

Broadcast Psi-calculi

Palle Raabjerg Johannes Åman Pohjola Tjark Weber

May 3, 2024

Abstract

We provide an Isabelle/HOL-Nominal formalisation of the definitions, theorems and proofs in the paper *Broadcast Psi-calculi with an Application to Wireless Protocols* by Borgström et al., which extends the Psi-calculi framework with primitives for broadcast communication in order to model wireless protocols.

1 Introduction

We provide an Isabelle/HOL-Nominal formalisation of the definitions, theorems and proofs in the paper *Broadcast Psi-calculi with an Application to Wireless Protocols* [4, 5], which extends the Psi-calculi framework [2, 3, 1] with primitives for broadcast communication in order to model wireless protocols.

The file `Broadcast_Thms.thy` contains a collection of the relevant definitions and theorems, with comments relating them directly to the paper.

2 Formalisation

```
theory Broadcast-Chain
imports Psi-Calculi.Chain
begin

lemma pair-perm-fresh-contr:
fixes a::'a and b::'a
assumes
  at: at TYPE('a)
and
  prems: b # pi (a, b) ∈ set pi
shows False
using prems
by(induct pi)
  (auto simp add: supp-list-cons fresh-list-nil supp-prod at-supp[OF at]
    fresh-list-cons fresh-prod at-fresh[OF at])
```

```

lemma pair-perm-fresh-contr':
  fixes  $a::'a$  and  $b::'a$ 
  assumes
     $at: at\ TYPE('a)$ 
  and
     $prems: a \# pi\ (a, b) \in set\ pi$ 
  shows False
  using prems
  by(induct pi)
    (auto simp add: supp-list-cons fresh-list-nil supp-prod at-supp[OF at]
      fresh-list-cons fresh-prod at-fresh[OF at])

lemma list-set-supp:
  fixes  $l :: ('d::fs-name)\ list$ 
  shows  $supp\ (set\ l) = (supp\ l :: name\ set)$ 
proof(induct l)
  case Nil then show ?case
    by (simp add: supp-list-nil supp-set-empty)
next
  case (Cons x xs) then show ?case
    using at-name-inst fs-name-inst pt-list-set-supp pt-name-inst by blast
qed

lemma name-set-supp:
  assumes finite a
  shows  $supp\ a = (a::name\ set)$ 
  using assms
  by(rule at-fin-set-supp[OF at-name-inst])

lemma supp-idem:
  fixes  $l :: ('d::fs-name)$ 
  shows  $supp((supp\ l)::name\ set) = (supp(l)::name\ set)$ 
proof –
  have  $f: finite((supp\ l)::name\ set)$ 
    by finite-guess
  show ?thesis
    by(rule name-set-supp[OF f])
qed

lemma fresh-supp:
  fixes  $a :: name$ 
  and  $X :: ('d::fs-name)$ 
  shows  $a \# ((supp\ X)::name\ set) = a \# X$ 
  by(simp add: fresh-def supp-idem)

lemma fresh-chain-supp:
  fixes  $A :: name\ list$ 
  and  $X :: ('d::fs-name)$ 
  shows  $A \#* ((supp\ X)::name\ set) = A \#* X$ 

```

```

unfolding fresh-star-def
by(simp add: fresh-supp)

lemma fresh-chain-fin-union:
  fixes  $X::('d::fs\text{-name set})$ 
    and  $Y::('d::fs\text{-name set})$ 
    and  $A::name\ list$ 
  assumes  $f1: finite\ X$ 
    and  $f2: finite\ Y$ 
  shows  $A\#\*(X\cup Y) = (A\#\*X \wedge A\#\*Y)$ 
  unfolding fresh-star-def
  apply(subst fresh-fin-union[OF pt-name-inst, OF at-name-inst, OF fs-name-inst,
OF f1, OF f2])
  by blast

lemma fresh-subset:
  fixes  $S :: name\ set$ 
    and  $S' :: name\ set$ 
    and  $a :: name$ 
  assumes  $a \# S$ 
    and  $S' \subseteq S$ 
    and finite S

shows  $a \# S'$ 
proof –
  have finite S' using  $\langle S' \subseteq S \rangle \langle finite\ S \rangle$ 
    by(rule Finite-Set.finite-subset)
  with assms show ?thesis
    by(auto simp add: fresh-def supp-subset Chain.name-list-supp name-set-supp)
qed

lemma fresh-subset':
  fixes  $S :: 'd::fs\text{-name set}$ 
    and  $S' :: 'd::fs\text{-name set}$ 
    and  $a :: name$ 
  assumes  $a \# S$ 
    and  $S' \subseteq S$ 
    and finite S

shows  $a \# S'$ 
proof –
  have finite S' using  $\langle S' \subseteq S \rangle \langle finite\ S \rangle$ 
    by(rule Finite-Set.finite-subset)
  have  $supp\ S' \subseteq ((supp\ S)::name\ set)$ 
    apply(rule Chain.supp-subset)
    by fact+
  with assms show ?thesis
    unfolding fresh-def
    by auto

```

qed

```
lemma fresh-star-subset':  
  fixes S :: 'd::fs-name set  
    and S' :: 'd::fs-name set  
    and A :: name list  
  assumes A #* S  
    and S'  $\subseteq$  S  
    and finite S
```

```
shows A #* S'  
  using assms  
  unfolding fresh-star-def  
  by(auto simp add: fresh-subset')
```

```
lemma fresh-star-subset:  
  fixes S :: name set  
    and S' :: name set  
    and A :: name list  
  assumes A #* S  
    and S'  $\subseteq$  S  
    and finite S
```

```
shows A #* S'  
  using assms  
  unfolding fresh-star-def  
  by(auto simp add: fresh-subset)
```

```
lemma times-set-fresh:  
  fixes a :: name  
    and S :: name list  
    and S' :: name list  
  assumes a # set S  
    and a # set S'  
  shows a # set S  $\times$  set S'  
  using assms  
proof(cases S)  
  case Nil then show ?thesis by(simp add: fresh-set-empty)  
next  
  case (Cons s Svec) then show ?thesis  
  proof(cases S')  
    case Nil then show ?thesis by(simp add: fresh-set-empty)  
  next  
    case (Cons s' S') then show ?thesis  
      using  $\langle S = s \# Svec \rangle$  assms  
      apply(subst fresh-cart-prod[OF pt-name-inst, OF pt-name-inst, OF fs-name-inst,  
OF fs-name-inst, OF at-name-inst])  
      by(simp+)  
qed
```

qed

lemma *times-set-fresh-star*:

fixes $A :: \text{name list}$

and $S :: \text{name list}$

and $S' :: \text{name list}$

assumes $A \#^* \text{set } S$

and $A \#^* \text{set } S'$

shows $A \#^* (\text{set } S \times \text{set } S')$

using *assms*

unfolding *fresh-star-def*

proof(*induct A*)

case Nil show ?case by simp

next

case(*Cons a A*)

have $a \# \text{set } S$ **and** $a \# \text{set } S'$ **using** *Cons* **by simp+**

then have $a \# (\text{set } S \times \text{set } S')$ **by**(*rule times-set-fresh*)

have $\forall x \in \text{set } A. (x \# \text{set } S)$ **and** $\forall x \in \text{set } A. (x \# \text{set } S')$ **using** *Cons*
by simp+

then have $\forall x \in \text{set } A. x \# \text{set } S \times \text{set } S'$ **using** *Cons* **by simp**

then show ?case **using** $\langle a \# (\text{set } S \times \text{set } S') \rangle$

by simp

qed

lemma *supp-list-set*:

fixes $M :: 'd :: \text{fs-name list}$

shows $(\text{supp } M) = ((\text{supp}(\text{set } M))::\text{name set})$

proof(*induct M*)

case Nil then show ?case by(*simp add: supp-set-empty supp-list-nil*)

next

case(*Cons m M*)

have $\text{lhs}: (\text{supp } (m \# M))::\text{name set} = \text{supp } m \cup \text{supp } M$ **by**(*simp add: supp-list-cons*)

have $\text{rhs}: (\text{supp } (\text{set } (m \# M)))::\text{name set} = \text{supp } m \cup \text{supp } M$

proof –

have $\text{supp } (\text{set } (m \# M)) = (\text{supp } (\text{set } [m] \cup \text{set } M))::\text{name set}$

by simp

moreover have $\dots = \text{supp } (\text{set } [m]) \cup \text{supp}(\text{set } M)$

apply(*rule supp-fin-union[OF pt-name-inst, OF at-name-inst, OF fs-name-inst]*)

by simp+

moreover have $\dots = \text{supp } m \cup \text{supp } M$

using *calculation(1) calculation(2) lhs list-set-supp* **by blast**

ultimately show ?thesis

by simp

qed

show ?case

unfolding *lhs rhs*

by(*rule refl*)

qed

```

lemma fresh-list-set:
  fixes  $M :: 'd :: fs\text{-name list}$ 
    and  $A :: name list$ 
  shows  $A \#* set M = A \#* M$ 
  unfolding fresh-star-def fresh-def supp-list-set
  by(rule refl)

lemma permSupp:
  fixes  $\Psi :: name prm$ 
    and  $\Psi' :: 'd :: fs\text{-name}$ 

shows  $(supp(\Psi \cdot \Psi')) :: name set \subseteq ((supp \Psi) \cup (supp \Psi'))$ 
proof -
  {
    fix  $x :: name$ 
    let  $?P = \lambda y. ([x, y]) \cdot \Psi \cdot ([x, y]) \cdot \Psi' \neq \Psi \cdot \Psi'$ 
    let  $?Q = \lambda y \Psi. ([x, y]) \cdot \Psi \neq \Psi$ 
    assume finite  $\{y. ?Q y \Psi'\}$ 
    moreover assume finite  $\{y. ?Q y \Psi\}$ 
    and infinite  $\{b. [(x, b)] \cdot \Psi \cdot \Psi' \neq \Psi \cdot \Psi'\}$ 
    from  $\langle i\text{finite} \{b. [(x, b)] \cdot \Psi \cdot \Psi' \neq \Psi \cdot \Psi'\} \rangle$  have infinite  $\{y. ?P(y)\}$ 
      by(subst cp1[symmetric, OF cp-pt-inst, OF pt-name-inst, OF at-name-inst])
    then have infinite( $\{y. ?P(y)\} - \{y. ?Q y \Psi\}$ )
      using  $\langle i\text{finite} \{y. ?Q y \Psi'\} \rangle \langle i\text{finite} \{y. ?Q y \Psi\} \rangle$ 
      by - (rule Diff-infinite-finite)
    ultimately have infinite( $(\{y. ?P(y)\} - \{y. ?Q y \Psi\}) - \{y. ?Q y \Psi'\}$ )
      by(rule Diff-infinite-finite)
    then have infinite( $\{y. ?P(y) \wedge \neg(?Q y \Psi) \wedge \neg(?Q y \Psi')\}$ ) by(simp add:
set-diff-eq)
    moreover have  $\{y. ?P(y) \wedge \neg(?Q y \Psi) \wedge \neg(?Q y \Psi')\} = \{\}$ 
      by auto
    ultimately have infinite  $\{\}$ 
      by - (drule Infinite-cong, auto)
    then have False by simp
  }
from this show ?thesis
  by(force simp add: supp-def)
qed

end
theory Broadcast-Frame
  imports Psi-Calculi.Frame
begin

locale assertionAux = Frame.assertionAux SCompose SImp SBottom SChanEq
for  $SCompose :: 'b :: fs\text{-name} \Rightarrow 'b \Rightarrow 'b$  (infixr  $\otimes$  80)
and  $SImp :: 'b \Rightarrow 'c :: fs\text{-name} \Rightarrow bool$  ( $- \vdash -$  [70, 70] 70)
and  $SBottom :: 'b$  ( $\perp$  90)
and  $SChanEq :: ('a :: fs\text{-name} \Rightarrow 'a \Rightarrow 'c)$  ( $- \leftrightarrow -$  [80, 80] 80)

```

```

+
fixes SOutCon :: 'a::fs-name  $\Rightarrow$  'a  $\Rightarrow$  'c (-  $\preceq$  - [80, 80] 80)
and SInCon :: 'a::fs-name  $\Rightarrow$  'a  $\Rightarrow$  'c (-  $\succeq$  - [80, 80] 80)

assumes statEqvt'''[eqvt]:  $\bigwedge p::\text{name prm. } p \cdot (M \preceq N) = (p \cdot M) \preceq (p \cdot N)$ 
and statEqvt''''[eqvt]:  $\bigwedge p::\text{name prm. } p \cdot (M \succeq N) = (p \cdot M) \succeq (p \cdot N)$ 

begin

lemma chanInConSupp:
fixes M :: 'a
and N :: 'a

shows (supp(M  $\succeq$  N)::name set)  $\subseteq$  ((supp M)  $\cup$  (supp N))
proof -
{
fix x::name
let ?P =  $\lambda y. ((x, y) \cdot M) \succeq [(x, y) \cdot N] \neq M \succeq N$ 
let ?Q =  $\lambda y M. ((x, y) \cdot M) \neq M$ 
assume finite {y. ?Q y N}
moreover assume finite {y. ?Q y M} and infinite {y. ?P(y)}
then have infinite({y. ?P(y)} - {y. ?Q y M}) by(rule Diff-infinite-finite)
ultimately have infinite(({y. ?P(y)} - {y. ?Q y M}) - {y. ?Q y N}) by(rule
Diff-infinite-finite)
then have infinite({y. ?P(y)  $\wedge$   $\neg$ (?Q y M)  $\wedge$   $\neg$  (?Q y N)}) by(simp add:
set-diff-eq)
moreover have {y. ?P(y)  $\wedge$   $\neg$ (?Q y M)  $\wedge$   $\neg$  (?Q y N)} = {} by auto
ultimately have infinite {} by(blast dest: Infinite-cong)
then have False by simp
}
then show ?thesis by(auto simp add: eqvts supp-def)
qed

lemma chanOutConSupp:
fixes M :: 'a
and N :: 'a

shows (supp(M  $\preceq$  N)::name set)  $\subseteq$  ((supp M)  $\cup$  (supp N))
proof -
{
fix x::name
let ?P =  $\lambda y. ((x, y) \cdot M) \preceq [(x, y) \cdot N] \neq M \preceq N$ 
let ?Q =  $\lambda y M. ((x, y) \cdot M) \neq M$ 
assume finite {y. ?Q y N}
moreover assume finite {y. ?Q y M} and infinite {y. ?P(y)}
then have infinite({y. ?P(y)} - {y. ?Q y M}) by(rule Diff-infinite-finite)
ultimately have infinite(({y. ?P(y)} - {y. ?Q y M}) - {y. ?Q y N}) by(rule
Diff-infinite-finite)
then have infinite({y. ?P(y)  $\wedge$   $\neg$ (?Q y M)  $\wedge$   $\neg$  (?Q y N)}) by(simp add:

```

```

set-diff-eq)
  moreover have  $\{y. ?P(y) \wedge \neg(?Q y M) \wedge \neg (?Q y N)\} = \{\}$  by auto
  ultimately have infinite  $\{\}$  by (blast dest: Infinite-cong)
  then have False by simp
}
then show ?thesis by (auto simp add: eqvts supp-def)
qed

```

lemma *freshInCon*[intro]:

```

fixes  $x :: \text{name}$ 
and  $M :: 'a$ 
and  $N :: 'a$ 

```

```

assumes  $x \# M$ 
and  $x \# N$ 

```

```

shows  $x \# M \succeq N$ 
using assms chanInConSupp
by (auto simp add: fresh-def)

```

lemma *freshInConChain*[intro]:

```

fixes  $\text{vec} :: \text{name list}$ 
and  $Xs :: \text{name set}$ 
and  $M :: 'a$ 
and  $N :: 'a$ 

```

```

shows  $[\text{vec} \#^* M; \text{vec} \#^* N] \implies \text{vec} \#^* (M \succeq N)$ 
and  $[Xs \#^* M; Xs \#^* N] \implies Xs \#^* (M \succeq N)$ 
by (auto simp add: fresh-star-def)

```

lemma *freshOutCon*[intro]:

```

fixes  $x :: \text{name}$ 
and  $M :: 'a$ 
and  $N :: 'a$ 

```

```

assumes  $x \# M$ 
and  $x \# N$ 

```

```

shows  $x \# M \preceq N$ 
using assms chanOutConSupp
by (auto simp add: fresh-def)

```

lemma *freshOutConChain*[intro]:

```

fixes  $\text{vec} :: \text{name list}$ 
and  $Xs :: \text{name set}$ 
and  $M :: 'a$ 
and  $N :: 'a$ 

```

```

shows  $[\text{vec} \#^* M; \text{vec} \#^* N] \implies \text{vec} \#^* (M \preceq N)$ 

```



```

and  $\llbracket Xs \#* M; Xs \#* N \rrbracket \implies Xs \#* (M \preceq N)$ 
by(auto simp add: fresh-star-def)

lemma chanOutConClosed:
  fixes  $\Psi :: 'b$ 
    and  $M :: 'a$ 
    and  $N :: 'a$ 
    and  $p :: \text{name prm}$ 

assumes  $\Psi \vdash M \preceq N$ 

shows  $(p \cdot \Psi) \vdash (p \cdot M) \preceq (p \cdot N)$ 
proof -
  from  $\langle \Psi \vdash M \preceq N \rangle$  have  $(p \cdot \Psi) \vdash p \cdot (M \preceq N)$ 
    by(rule statClosed)
  then show ?thesis by(auto simp add: eqts)
qed

lemma chanInConClosed:
  fixes  $\Psi :: 'b$ 
    and  $M :: 'a$ 
    and  $N :: 'a$ 
    and  $p :: \text{name prm}$ 

assumes  $\Psi \vdash M \succeq N$ 

shows  $(p \cdot \Psi) \vdash (p \cdot M) \succeq (p \cdot N)$ 
proof -
  from  $\langle \Psi \vdash M \succeq N \rangle$  have  $(p \cdot \Psi) \vdash p \cdot (M \succeq N)$ 
    by(rule statClosed)
  then show ?thesis by(auto simp add: eqts)
qed

end

locale assertion = assertionAux SCompose SImp SBottom SChanEq SOutCon SInCon
  Con + assertion SCompose SImp SBottom SChanEq
  for SCompose ::  $'b::\text{fs-name} \Rightarrow 'b \Rightarrow 'b$ 
  and SImp ::  $'b \Rightarrow 'c::\text{fs-name} \Rightarrow \text{bool}$ 
  and SBottom ::  $'b$ 
  and SChanEq ::  $'a::\text{fs-name} \Rightarrow 'a \Rightarrow 'c$ 
  and SOutCon ::  $'a::\text{fs-name} \Rightarrow 'a \Rightarrow 'c$ 
  and SInCon ::  $'a::\text{fs-name} \Rightarrow 'a \Rightarrow 'c$  +

  assumes chanOutConSupp:  $SImp \Psi (SOutCon M N) \implies (((\text{supp } N)::\text{name set})$ 
     $\subseteq ((\text{supp } M)::\text{name set}))$ 
  and chanInConSupp:  $SImp \Psi (SInCon N M) \implies (((\text{supp } N)::\text{name set})$ 
     $\subseteq ((\text{supp } M)::\text{name set}))$ 

```

begin

notation $SOutCon$ ($- \preceq -$ $[90, 90]$ 90)

notation $SInCon$ ($- \succeq -$ $[90, 90]$ 90)

end

end

theory *Semantics*

imports *Broadcast-Chain Broadcast-Frame*

begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Semantics* from [1]. The nominal datatypes (*'a, 'b, 'c*) *residual* and *'a action* have been extended with constructors for broadcast input and output. This leads to a different semantics.

nominal-datatype (*'a, 'b, 'c*) *boundOutput* =
 $BOut$ *'a::fs-name* (*'a, 'b::fs-name, 'c::fs-name*) *psi* ($- \prec'' -$ $[110, 110]$ 110)
 | $BStep$ $\langle name \rangle$ (*'a, 'b, 'c*) *boundOutput* ($(\nu-)$ - $[110, 110]$ 110)

primrec $BOresChain$:: *name list* \Rightarrow (*'a::fs-name, 'b::fs-name, 'c::fs-name*) *bound-Output* \Rightarrow

(*'a, 'b, 'c*) *boundOutput*

where

$Base$: $BOresChain$ \square $B = B$

 | $Step$: $BOresChain$ ($x\#xs$) $B = (\nu x)(BOresChain$ xs $B)$

abbreviation

$BOresChainJudge$ ($(\nu*-)$ - $[80, 80]$ 80) **where** $(\nu*xvec)B \equiv BOresChain$ $xvec$ B

lemma $BOresChainEqvt[eqvt]$:

fixes $perm$:: *name prm*

and lst :: *name list*

and B :: (*'a::fs-name, 'b::fs-name, 'c::fs-name*) *boundOutput*

shows $perm \cdot ((\nu*xvec)B) = (\nu*(perm \cdot xvec))(\mathit{perm} \cdot B)$

by(*induct xvec*) *auto*

lemma $BOresChainSimps[simp]$:

fixes $xvec$:: *name list*

and N :: *'a::fs-name*

and P :: (*'a, 'b::fs-name, 'c::fs-name*) *psi*

and N' :: *'a*

and P' :: (*'a, 'b, 'c*) *psi*

and B :: (*'a, 'b, 'c*) *boundOutput*

and B' :: (*'a, 'b, 'c*) *boundOutput*

shows $((\nu*xvec)N \prec' P = N' \prec' P') = (xvec = \square \wedge N = N' \wedge P = P')$

and $(N' \prec' P' = (\nu*xvec)N \prec' P) = (xvec = \square \wedge N = N' \wedge P = P')$

and $(N' \prec' P' = N \prec' P) = (N = N' \wedge P = P')$
and $(\langle \nu * xvec \rangle B = \langle \nu * xvec \rangle B') = (B = B')$
by(*induct xvec*) (*auto simp add: boundOutput.inject alpha*)

lemma *outputFresh[simp]*:

fixes $Xs \quad :: \textit{name set}$
and $xvec \quad :: \textit{name list}$
and $N \quad \quad :: \textit{'a::fs-name}$
and $P \quad \quad :: (\textit{'a}, \textit{'b::fs-name}, \textit{'c::fs-name}) \textit{psi}$

shows $(Xs \#* (N \prec' P)) = ((Xs \#* N) \wedge (Xs \#* P))$
and $(xvec \#* (N \prec' P)) = ((xvec \#* N) \wedge (xvec \#* P))$
by(*auto simp add: fresh-star-def*)

lemma *boundOutputFresh*:

fixes $x \quad \quad :: \textit{name}$
and $xvec \quad :: \textit{name list}$
and $B \quad \quad :: (\textit{'a::fs-name}, \textit{'b::fs-name}, \textit{'c::fs-name}) \textit{boundOutput}$

shows $(x \# (\langle \nu * xvec \rangle B)) = (x \in \textit{set } xvec \vee x \# B)$
by (*induct xvec*) (*simp-all add: abs-fresh*)

lemma *boundOutputFreshSet*:

fixes $Xs \quad \quad :: \textit{name set}$
and $xvec \quad :: \textit{name list}$
and $B \quad \quad :: (\textit{'a::fs-name}, \textit{'b::fs-name}, \textit{'c::fs-name}) \textit{boundOutput}$
and $yvec \quad :: \textit{name list}$
and $x \quad \quad :: \textit{name}$

shows $Xs \#* (\langle \nu * xvec \rangle B) = (\forall x \in Xs. x \in \textit{set } xvec \vee x \# B)$
and $yvec \#* (\langle \nu * xvec \rangle B) = (\forall x \in (\textit{set } yvec). x \in \textit{set } xvec \vee x \# B)$
and $Xs \#* (\langle \nu x \rangle B) = Xs \#* [x].B$
and $xvec \#* (\langle \nu x \rangle B) = xvec \#* [x].B$
by(*simp add: fresh-star-def boundOutputFresh*)+

lemma *BOresChainSupp*:

fixes $xvec \quad :: \textit{name list}$
and $B \quad \quad :: (\textit{'a::fs-name}, \textit{'b::fs-name}, \textit{'c::fs-name}) \textit{boundOutput}$

shows $(\textit{supp}(\langle \nu * xvec \rangle B)::\textit{name set}) = (\textit{supp } B) - (\textit{supp } xvec)$
by(*induct xvec*)
(auto simp add: boundOutput.supp supp-list-nil supp-list-cons abs-supp supp-atm)

lemma *boundOutputFreshSimps[simp]*:

fixes $Xs \quad \quad :: \textit{name set}$
and $xvec \quad :: \textit{name list}$
and $B \quad \quad :: (\textit{'a::fs-name}, \textit{'b::fs-name}, \textit{'c::fs-name}) \textit{boundOutput}$
and $yvec \quad :: \textit{name list}$
and $x \quad \quad :: \textit{name}$

shows $Xs \#* xvec \implies (Xs \#* (\nu*xvec)B) = (Xs \#* B)$
and $yvec \#* xvec \implies yvec \#* (\nu*xvec)B = yvec \#* B$
and $xvec \#* (\nu*xvec)B$
and $x \# xvec \implies x \# (\nu*xvec)B = x \# B$
apply(*simp add: boundOutputFreshSet*) **apply**(*force simp add: fresh-star-def*
name-list-supp fresh-def)
apply(*simp add: boundOutputFreshSet*) **apply**(*force simp add: fresh-star-def*
name-list-supp fresh-def)
apply(*simp add: boundOutputFreshSet*)
by(*simp add: BOresChainSupp fresh-def*)

lemma *boundOutputChainAlpha*:

fixes $p \quad :: \text{name prm}$
and $xvec \quad :: \text{name list}$
and $B \quad :: ('a::\text{fs-name}, 'b::\text{fs-name}, 'c::\text{fs-name}) \text{boundOutput}$
and $yvec \quad :: \text{name list}$

assumes $xvecFreshB: (p \cdot xvec) \#* B$
and $S: \text{set } p \subseteq \text{set } xvec \times \text{set } (p \cdot xvec)$
and $(\text{set } xvec) \subseteq (\text{set } yvec)$

shows $(\nu*yvec)B = (\nu*(p \cdot yvec))(p \cdot B)$

proof –

note *pt-name-inst at-name-inst S*
moreover from $\langle (\text{set } xvec) \subseteq (\text{set } yvec) \rangle$ **have** $\text{set } xvec \#* (\nu*yvec)B$
by(*force simp add: boundOutputFreshSet*)
moreover from $xvecFreshB \langle (\text{set } xvec) \subseteq (\text{set } yvec) \rangle$ **have** $\text{set } (p \cdot xvec) \#*$
 $(\nu*yvec)B$
by (*simp add: boundOutputFreshSet*) (*simp add: fresh-star-def*)
ultimately have $(\nu*yvec)B = p \cdot (\nu*yvec)B$
by (*rule pt-freshs-freshs [symmetric]*)
then show *?thesis* **by**(*simp add: eqvts*)

qed

lemma *boundOutputChainAlpha'*:

fixes $p \quad :: \text{name prm}$
and $xvec \quad :: \text{name list}$
and $B \quad :: ('a::\text{fs-name}, 'b::\text{fs-name}, 'c::\text{fs-name}) \text{boundOutput}$
and $yvec \quad :: \text{name list}$
and $zvec \quad :: \text{name list}$

assumes $xvecFreshB: xvec \#* B$
and $S: \text{set } p \subseteq \text{set } xvec \times \text{set } yvec$
and $yvec \#* (\nu*zvec)B$

shows $(\nu*zvec)B = (\nu*(p \cdot zvec))(p \cdot B)$

proof –

note *pt-name-inst at-name-inst S* $\langle yvec \#* (\nu*zvec)B \rangle$

moreover from $xvecFreshB$ **have** $set(xvec) \#* (\nu * xvec) B$
by (*simp add: boundOutputFreshSet*) (*simp add: fresh-star-def*)
ultimately have $(\nu * xvec) B = p \cdot (\nu * xvec) B$
by(*auto intro: pt-freshs-freshs [symmetric]*)
then show *?thesis* **by**(*simp add: eqvts*)
qed

lemma *boundOutputChainAlpha''*:

fixes $p :: name\ prm$
and $xvec :: name\ list$
and $M :: 'a::fs-name$
and $P :: ('a::fs-name, 'b::fs-name, 'c::fs-name)\ psi$
and $yvec :: name\ list$

assumes $(p \cdot xvec) \#* M$
and $(p \cdot xvec) \#* P$
and $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$
and $(set\ xvec) \subseteq (set\ yvec)$

shows $(\nu * yvec) M \prec' P = (\nu * (p \cdot yvec)) (p \cdot M) \prec' (p \cdot P)$
using *assms*
by(*subst boundOutputChainAlpha*) *auto*

lemma *boundOutputChainSwap*:

fixes $x :: name$
and $y :: name$
and $N :: 'a::fs-name$
and $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$
and $xvec :: name\ list$

assumes $y \# N$
and $y \# P$
and $x \in (set\ xvec)$

shows $(\nu * xvec) N \prec' P = (\nu * ((x, y) \cdot xvec)) ((x, y) \cdot N) \prec' ((x, y) \cdot P)$

proof(*cases x=y*)

assume $x=y$

then show *?thesis* **by** *simp*

next

assume $x \neq y$

with *assms* **show** *?thesis*

by(*auto simp add: calc-atm intro: boundOutputChainAlpha''*[**where** $xvec=[x]$])

qed

lemma *alphaBoundOutput*:

fixes $x :: name$

and $y :: name$

and $B :: ('a::fs-name, 'b::fs-name, 'c::fs-name)\ boundOutput$

assumes $y \# B$

shows $(\nu x)B = (\nu y)((x, y) \cdot B)$

using *assms*

by(*auto simp add: boundOutput.inject alpha fresh-left calc-atm*)

lemma *boundOutputEqFresh*:

fixes $B :: ('a::fs-name, 'b::fs-name, 'c::fs-name) \text{boundOutput}$

and $C :: ('a, 'b, 'c) \text{boundOutput}$

and $x :: \text{name}$

and $y :: \text{name}$

assumes $(\nu x)B = (\nu y)C$

and $x \# B$

shows $y \# C$

using *assms*

by(*auto simp add: boundOutput.inject alpha fresh-left calc-atm*)

lemma *boundOutputEqSupp*:

fixes $B :: ('a::fs-name, 'b::fs-name, 'c::fs-name) \text{boundOutput}$

and $C :: ('a, 'b, 'c) \text{boundOutput}$

and $x :: \text{name}$

and $y :: \text{name}$

assumes $(\nu x)B = (\nu y)C$

and $x \in \text{supp } B$

shows $y \in \text{supp } C$

using *assms*

apply(*clarsimp simp add: boundOutput.inject alpha fresh-left calc-atm*)

apply(*drule pt-set-bij2[where pi=[(x, y)], OF pt-name-inst, OF at-name-inst]*)

by(*auto simp add: eqvts calc-atm*)

lemma *boundOutputChainEq*:

fixes $xvec :: \text{name list}$

and $B :: ('a::fs-name, 'b::fs-name, 'c::fs-name) \text{boundOutput}$

and $yvec :: \text{name list}$

and $B' :: ('a, 'b, 'c) \text{boundOutput}$

assumes $(\nu * xvec)B = (\nu * yvec)B'$

and $xvec \#* yvec$

and $\text{length } xvec = \text{length } yvec$

shows $\exists p. (\text{set } p) \subseteq (\text{set } xvec) \times \text{set } (yvec) \wedge \text{distinctPerm } p \wedge B = p \cdot B' \wedge$
 $(\text{set } (\text{map } \text{fst } p)) \subseteq (\text{supp } B) \wedge xvec \#* B' \wedge yvec \#* B$

proof –

obtain n **where** $n = \text{length } xvec$ **by** *auto*

with *assms* **show** *?thesis*

```

proof(induct n arbitrary: xvec yvec B B')
  case(0 xvec yvec B B')
    have Eq:  $(\nu * xvec)B = (\nu * yvec)B'$  by fact
    from  $\langle 0 = \text{length } xvec \rangle$  have  $xvec = []$  by auto
    moreover with  $\langle \text{length } xvec = \text{length } yvec \rangle$  have  $yvec = []$ 
      by(cases yvec) auto
    ultimately show ?case using Eq
      by(simp add: boundOutput.inject)
next
  case(Suc n xvec yvec B B')
    from  $\langle \text{Suc } n = \text{length } xvec \rangle$ 
    obtain  $x \ xvec'$  where  $xvec = x \# xvec'$  and  $\text{length } xvec' = n$ 
      by(cases xvec) auto
    from  $\langle (\nu * xvec)B = (\nu * yvec)B' \rangle \langle xvec = x \# xvec' \rangle \langle \text{length } xvec = \text{length } yvec \rangle$ 
    obtain  $y \ yvec'$  where  $(\nu * (x \# xvec'))B = (\nu * (y \# yvec'))B'$ 
      and  $yvec = y \# yvec'$  and  $\text{length } xvec' = \text{length } yvec'$ 
      by(cases yvec) auto
    then have EQ:  $(\nu x)((\nu * xvec')B) = (\nu y)((\nu * yvec')B')$ 
      by simp
    from  $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle xvec \#* yvec \rangle$ 
    have  $x \neq y$  and  $xvec' \#* yvec'$  and  $x \# yvec'$  and  $y \# xvec'$ 
      by auto
    have IH:  $\bigwedge yvec \ yvec' \ B \ B'. \llbracket (\nu * xvec)(B::('a::fs-name, 'b::fs-name, 'c::fs-name) \text{boundOutput}) = (\nu * yvec)B'; xvec \#* yvec; \text{length } xvec = \text{length } yvec; n = \text{length } xvec \rrbracket \implies \exists p. (\text{set } p) \subseteq (\text{set } xvec) \times (\text{set } yvec) \wedge \text{distinctPerm } p \wedge B = p \cdot B' \wedge \text{set}(\text{map } \text{fst } p) \subseteq \text{supp } B \wedge xvec \#* B' \wedge yvec \#* B$ 
      by fact
    from EQ  $\langle x \neq y \rangle$  have EQ':  $(\nu * xvec')B = ([x, y]) \cdot ((\nu * yvec')B')$ 
      and xFreshB':  $x \# ((\nu * yvec')B')$ 
      and yFreshB:  $y \# ((\nu * xvec')B)$ 
      by(metis boundOutput.inject alpha)+
    from xFreshB'  $\langle x \# yvec' \rangle$  have  $x \# B'$ 
      by(auto simp add: boundOutputFresh) (simp add: fresh-def name-list-supp)+
    from yFreshB  $\langle y \# xvec' \rangle$  have  $y \# B$ 
      by(auto simp add: boundOutputFresh) (simp add: fresh-def name-list-supp)+
    show ?case
    proof(cases x \# ((\nu * xvec')B))
      assume xFreshB:  $x \# ((\nu * xvec')B)$ 
      with EQ have yFreshB':  $y \# ((\nu * yvec')B')$ 
        by(rule boundOutputEqFresh)
      with xFreshB' EQ' have  $(\nu * xvec')B = (\nu * yvec')B'$ 
        by(simp)
      with  $\langle xvec' \#* yvec' \rangle \langle \text{length } xvec' = \text{length } yvec' \rangle \langle \text{length } xvec' = n \rangle$  IH
      obtain  $p$  where  $S: (\text{set } p) \subseteq (\text{set } xvec') \times (\text{set } yvec') \text{ and } \text{distinctPerm } p$ 
and  $B = p \cdot B'$ 
      and  $\text{set}(\text{map } \text{fst } p) \subseteq \text{supp } B$  and  $xvec' \#* B'$  and  $yvec' \#* B$ 
      by blast
    from  $S$  have  $(\text{set } p) \subseteq \text{set}(x \# xvec') \times \text{set}(y \# yvec')$  by auto
    moreover note  $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle \text{distinctPerm } p \rangle \langle B = p$ 

```

$\cdot B'$
 $\langle xvec' \#* B' \rangle \langle x \# B' \rangle \langle x \# B' \rangle \langle yvec' \#* B \rangle \langle y \# B \rangle \langle set(map fst p) \subseteq supp B \rangle$

ultimately show *?case by auto*
next
assume $\neg(x \# (\nu*xvec')B)$
then have $xSuppB: x \in supp((\nu*xvec')B)$
by(*simp add: fresh-def*)
with EQ **have** $ySuppB': y \in supp((\nu*yvec')B')$
by(*rule boundOutputEqSupp*)
then have $y \# yvec'$
by(*induct yvec'*) (*auto simp add: boundOutput.supp abs-supp*)
with $\langle x \# yvec' \rangle EQ'$ **have** $(\nu*xvec')B = (\nu*yvec')([\langle x, y \rangle] \cdot B')$
by(*simp add: eqvts*)
with $\langle xvec' \#* yvec' \rangle \langle length xvec' = length yvec' \rangle \langle length xvec' = n \rangle IH$
obtain p **where** $S: (set p) \subseteq (set xvec') \times (set yvec')$ **and** *distinctPerm p*
and $B = p \cdot [\langle x, y \rangle] \cdot B'$
and $set(map fst p) \subseteq supp B$ **and** $xvec' \#* ([\langle x, y \rangle] \cdot B')$ **and** $yvec' \#* B$
by *blast*

from $xSuppB$ **have** $x \# xvec'$
by(*induct xvec'*) (*auto simp add: boundOutput.supp abs-supp*)
with $\langle x \# yvec' \rangle \langle y \# xvec' \rangle \langle y \# yvec' \rangle S$ **have** $x \# p$ **and** $y \# p$
apply(*induct p*)
by(*auto simp add: name-list-supp*) (*auto simp add: fresh-def*)
from S **have** $(set([\langle x, y \rangle] \# p)) \subseteq (set(x\#xvec')) \times (set(y\#yvec'))$
by *force*
moreover from $\langle x \neq y \rangle \langle x \# p \rangle \langle y \# p \rangle S$ *distinctPerm p*
have *distinctPerm([\langle x, y \rangle] \# p)* **by** *simp*
moreover from $\langle B = p \cdot [\langle x, y \rangle] \cdot B' \rangle \langle x \# p \rangle \langle y \# p \rangle$ **have** $B = [\langle x, y \rangle] \cdot p$
 $\cdot B'$
by(*subst perm-compose*) *simp*
then have $B = ([\langle x, y \rangle] \# p) \cdot B'$ **by** *simp*
moreover from $\langle xvec' \#* ([\langle x, y \rangle] \cdot B') \rangle$ **have** $([\langle x, y \rangle] \cdot xvec') \#* ([\langle x, y \rangle] \cdot B')$
 $[\langle x, y \rangle] \cdot B'$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# xvec' \rangle \langle y \# xvec' \rangle \langle x \# B' \rangle$ **have** $(x\#xvec') \#* B'$ **by** *simp*
moreover from $\langle y \# B \rangle \langle yvec' \#* B \rangle$ **have** $(y\#yvec') \#* B$ **by** *simp*
moreover from $\langle set(map fst p) \subseteq supp B \rangle xSuppB \langle x \# xvec' \rangle$
have $set(map fst([\langle x, y \rangle] \# p)) \subseteq supp B$
by(*simp add: BOrChainSupp*)
ultimately show *?case using* $\langle xvec=x\#xvec' \rangle \langle yvec=y\#yvec' \rangle$
by *metis*

qed
qed
qed

lemma *boundOutputChainEqLength*:


```

fixes  $xvec :: name\ list$ 
and  $M :: 'a::fs-name$ 
and  $P :: ('a, 'b::fs-name, 'c::fs-name) psi$ 
and  $yvec :: name\ list$ 
and  $N :: 'a::fs-name$ 
and  $Q :: ('a, 'b::fs-name, 'c::fs-name) psi$ 

assumes  $(\nu*xvec)M \prec' P = (\nu*yvec)N \prec' Q$ 

shows  $length\ xvec = length\ yvec$ 
proof -
  obtain  $n$  where  $n = length\ xvec$  by auto
  with assms show ?thesis
  proof(induct n arbitrary: xvec yvec M P N Q)
    case( $0\ xvec\ yvec\ M\ P\ N\ Q$ )
      from  $\langle 0 = length\ xvec \rangle$  have  $xvec = []$  by auto
      moreover with  $\langle (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' Q \rangle$  have  $yvec = []$ 
        by(cases yvec) auto
      ultimately show ?case by simp
  next
    case( $Suc\ n\ xvec\ yvec\ M\ P\ N\ Q$ )
      from  $\langle Suc\ n = length\ xvec \rangle$ 
      obtain  $x\ xvec'$  where  $xvec = x\#\ xvec'$  and  $length\ xvec' = n$ 
        by(cases xvec) auto
      from  $\langle (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' Q \rangle$   $\langle xvec = x\#\ xvec' \rangle$ 
      obtain  $y\ yvec'$  where  $(\nu*(x\#\ xvec'))M \prec' P = (\nu*(y\#\ yvec'))N \prec' Q$ 
        and  $yvec = y\#\ yvec'$ 
        by(cases yvec) auto
      then have  $EQ: (\nu x)((\nu*xvec')M \prec' P) = (\nu y)((\nu*yvec')N \prec' Q)$ 
        by simp
      have  $IH: \bigwedge xvec\ yvec\ M\ P\ N\ Q. [(\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q::('a, 'b,$ 
'c) psi); n = length xvec]] \implies length xvec = length yvec
        by fact
      show ?case
      proof(cases x = y)
        assume  $x = y$ 
        with  $EQ$  have  $(\nu*xvec')M \prec' P = (\nu*yvec')N \prec' Q$ 
          by(simp add: alpha boundOutput.inject)
        with  $IH\ \langle length\ xvec' = n \rangle$  have  $length\ xvec' = length\ yvec'$ 
          by blast
        with  $\langle xvec = x\#\ xvec' \rangle\ \langle yvec = y\#\ yvec' \rangle$ 
        show ?case by simp
      next
        assume  $x \neq y$ 
        with  $EQ$  have  $(\nu*xvec')M \prec' P = [(x, y)] \cdot (\nu*yvec')N \prec' Q$ 
          by(simp add: alpha boundOutput.inject)
        then have  $(\nu*xvec')M \prec' P = (\nu*([(x, y)] \cdot yvec'))([[(x, y)] \cdot N] \prec' ([[(x, y)]$ 
\cdot Q])
          by(simp add: eqvts)

```

```

with IH ⟨length xvec' = n⟩ have length xvec' = length ((x, y)] · yvec')
  by blast
then have length xvec' = length yvec'
  by simp
with ⟨xvec = x#xvec'⟩ ⟨yvec=y#yvec'⟩
show ?case by simp
qed
qed
qed

```

lemma *boundOutputChainEq'*:

```

fixes xvec :: name list
and M    :: 'a::fs-name
and P    :: ('a, 'b::fs-name, 'c::fs-name) psi
and yvec :: name list
and N    :: 'a
and Q    :: ('a::fs-name, 'b::fs-name, 'c::fs-name) psi

```

```

assumes (ν*xvec)M <' P = (ν*yvec)N <' Q
and xvec #* yvec

```

```

shows ∃ p. (set p) ⊆ (set xvec) × set (yvec) ∧ distinctPerm p ∧ M = p · N ∧ P
= p · Q ∧ xvec #* N ∧ xvec #* Q ∧ yvec #* M ∧ yvec #* P
using assms boundOutputChainEq boundOutputChainEqLength by fastforce

```

lemma *boundOutputChainEq''*:

```

fixes xvec :: name list
and M    :: 'a::fs-name
and P    :: ('a, 'b::fs-name, 'c::fs-name) psi
and yvec :: name list
and N    :: 'a
and Q    :: ('a::fs-name, 'b::fs-name, 'c::fs-name) psi

```

```

assumes (ν*xvec)M <' P = (ν*yvec)N <' Q
and xvec #* yvec
and distinct xvec
and distinct yvec

```

```

obtains p where (set p) ⊆ (set xvec) × set (p · xvec) and distinctPerm p and
yvec = p · xvec and N = p · M and Q = p · P and xvec #* N and xvec #* Q
and (p · xvec) #* M and (p · xvec) #* P

```

proof –

```

assume ∧ p. [set p ⊆ set xvec × set (p · xvec); distinctPerm p; yvec = p · xvec;
N = p · M; Q = p · P; xvec #* N; xvec #* Q; (p · xvec) #* M; (p · xvec) #* P]
⇒ thesis

```

moreover obtain *n* **where** *n* = length xvec **by** auto

with *assms* **have** ∃ p. (set p) ⊆ (set xvec) × set (yvec) ∧ distinctPerm p ∧ yvec

$= p \cdot xvec \wedge N = p \cdot M \wedge Q = p \cdot P \wedge xvec \#* N \wedge xvec \#* Q \wedge (p \cdot xvec) \#* M$
 $\wedge (p \cdot xvec) \#* P$
proof(*induct n arbitrary: xvec yvec M P N Q*)
case(0 *xvec yvec M P N Q*)
have $Eq: (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' Q$ **by fact**
from $\langle 0 = \text{length } xvec \rangle$ **have** $xvec = []$ **by auto**
moreover with Eq **have** $yvec = []$
by(*cases yvec*) **auto**
ultimately show *?case using Eq*
by(*simp add: boundOutput.inject*)
next
case(*Suc n xvec yvec M P N Q*)
from $\langle \text{Suc } n = \text{length } xvec \rangle$
obtain x $xvec'$ **where** $xvec = x\#xvec'$ **and** $\text{length } xvec' = n$
by(*cases xvec*) **auto**
from $\langle (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' Q \rangle$ $\langle xvec = x\#xvec' \rangle$
obtain y $yvec'$ **where** $(\nu*(x\#xvec'))M \prec' P = (\nu*(y\#yvec'))N \prec' Q$
and $yvec = y\#yvec'$
by(*cases yvec*) **auto**
then have $EQ: (\nu x)((\nu*xvec')M \prec' P) = (\nu y)((\nu*yvec')N \prec' Q)$
by simp
from $\langle xvec = x\#xvec' \rangle$ $\langle yvec = y\#yvec' \rangle$ $\langle xvec \#* yvec \rangle$
have $x \neq y$ **and** $xvec' \#* yvec'$ **and** $x \# yvec'$ **and** $y \# xvec'$
by auto
from $\langle \text{distinct } xvec \rangle$ $\langle \text{distinct } yvec \rangle$ $\langle xvec = x\#xvec' \rangle$ $\langle yvec = y\#yvec' \rangle$ **have** $x \#$
 $xvec'$ **and** $y \# yvec'$ **and** $\text{distinct } xvec'$ **and** $\text{distinct } yvec'$
by simp+
have $IH: \bigwedge xvec yvec M P N Q. \llbracket (\nu*xvec)(M::'a) \prec' (P::('a, 'b, 'c) psi) =$
 $(\nu*yvec)N \prec' Q; xvec \#* yvec; \text{distinct } xvec; \text{distinct } yvec; n = \text{length } xvec \rrbracket \implies$
 $\exists p. (\text{set } p) \subseteq (\text{set } xvec) \times (\text{set } yvec) \wedge \text{distinctPerm } p \wedge yvec = p \cdot xvec \wedge N =$
 $p \cdot M \wedge Q = p \cdot P \wedge xvec \#* N \wedge xvec \#* Q \wedge (p \cdot xvec) \#* M \wedge (p \cdot xvec) \#* P$
by fact
from $EQ \langle x \neq y \rangle$ $\langle x \# yvec' \rangle$ $\langle y \# yvec' \rangle$ $\langle y \# xvec' \rangle$ $\langle x \# xvec' \rangle$ **have** $(\nu*xvec')M$
 $\prec' P = (\nu*yvec')(\llbracket (x, y) \rrbracket \cdot N) \prec' (\llbracket (x, y) \rrbracket \cdot Q)$ **and** $x \# N$ **and** $x \# Q$ **and** $y \# M$
and $y \# P$
apply –
apply(*simp add: boundOutput.inject alpha eqvts*)
apply(*simp add: boundOutput.inject alpha eqvts*)
apply(*simp add: boundOutput.inject alpha eqvts*)
by(*simp add: boundOutput.inject alpha' eqvts*)+
with $\langle xvec' \#* yvec' \rangle$ $\langle \text{distinct } xvec' \rangle$ $\langle \text{distinct } yvec' \rangle$ $\langle \text{length } xvec' = n \rangle$ IH
obtain p **where** $S: (\text{set } p) \subseteq (\text{set } xvec') \times (\text{set } yvec')$ **and** $\text{distinctPerm } p$ **and**
 $yvec' = p \cdot xvec'$ **and** $\llbracket (x, y) \rrbracket \cdot N = p \cdot M$ **and** $\llbracket (x, y) \rrbracket \cdot Q = p \cdot P$ **and** $xvec'$
 $\#* (\llbracket (x, y) \rrbracket \cdot N)$ **and** $xvec' \#* (\llbracket (x, y) \rrbracket \cdot Q)$ **and** $yvec' \#* M$ **and** $yvec' \#* P$
by metis
from S **have** $\text{set}(\llbracket (x, y) \rrbracket \cdot p) \subseteq \text{set}(x\#xvec') \times \text{set}(y\#yvec')$ **by auto**
moreover from $\langle x \# xvec' \rangle$ $\langle x \# yvec' \rangle$ $\langle y \# xvec' \rangle$ $\langle y \# yvec' \rangle$ S **have** $x \# p$ **and**
 $y \# p$
apply(*induct p*)

by(*auto simp add: fresh-prod name-list-supp*) (*auto simp add: fresh-def*)
with $S \langle \text{distinctPerm } p \rangle \langle x \neq y \rangle$ **have** $\text{distinctPerm}((x, y)\#p)$ **by** *auto*
moreover from $\langle yvec' = p \cdot xvec' \rangle \langle x \# p \rangle \langle y \# p \rangle \langle x \# xvec' \rangle \langle y \# xvec' \rangle$ **have**
 $(y\#yvec') = ((x, y)\#p) \cdot (x\#xvec')$
by(*simp add: eqts calc-atm perm-compose freshChainSimps*)
moreover from $\langle [(x, y)] \cdot N = p \cdot M \rangle$
have $([(x, y)] \cdot [(x, y)] \cdot N) = [(x, y)] \cdot p \cdot M$
by(*simp add: pt-bij*)
then have $N = ((x, y)\#p) \cdot M$ **by** *simp*
moreover from $\langle [(x, y)] \cdot Q = p \cdot P \rangle$
have $([(x, y)] \cdot [(x, y)] \cdot Q) = [(x, y)] \cdot p \cdot P$
by(*simp add: pt-bij*)
then have $Q = ((x, y)\#p) \cdot P$ **by** *simp*
moreover from $\langle xvec' \#* [(x, y)] \cdot N \rangle$ **have** $([(x, y)] \cdot xvec') \#* [(x, y)] \cdot$
 $[(x, y)] \cdot N$
by(*subst fresh-star-bij*)
with $\langle x \# xvec' \rangle \langle y \# xvec' \rangle$ **have** $xvec' \#* N$ **by** *simp*
with $\langle x \# N \rangle$ **have** $(x\#xvec') \#* N$ **by** *simp*
moreover from $\langle xvec' \#* [(x, y)] \cdot Q \rangle$ **have** $([(x, y)] \cdot xvec') \#* [(x, y)] \cdot$
 $[(x, y)] \cdot Q$
by(*subst fresh-star-bij*)
with $\langle x \# xvec' \rangle \langle y \# xvec' \rangle$ **have** $xvec' \#* Q$ **by** *simp*
with $\langle x \# Q \rangle$ **have** $(x\#xvec') \#* Q$ **by** *simp*
moreover from $\langle y \# M \rangle \langle yvec' \#* M \rangle$ **have** $(y\#yvec') \#* M$ **by** *simp*
moreover from $\langle y \# P \rangle \langle yvec' \#* P \rangle$ **have** $(y\#yvec') \#* P$ **by** *simp*
ultimately show $?case$ **using** $\langle xvec = x\#xvec' \rangle \langle yvec = y\#yvec' \rangle$
by *metis*
qed
ultimately show $?thesis$ **by** *blast*
qed

lemma *boundOutputEqSupp'*:

fixes $x :: \text{name}$
and $xvec :: \text{name list}$
and $M :: 'a::\text{fs-name}$
and $P :: ('a, 'b::\text{fs-name}, 'c::\text{fs-name}) \text{psi}$
and $y :: \text{name}$
and $yvec :: \text{name list}$
and $N :: 'a$
and $Q :: ('a, 'b, 'c) \text{psi}$

assumes $Eq: (\nu x)(\nu *xvec)M \prec' P = (\nu y)(\nu *yvec)N \prec' Q$

and $x \neq y$
and $x \# yvec$
and $x \# xvec$
and $y \# xvec$
and $y \# yvec$
and $xvec \#* yvec$

and $x \in \text{supp } M$
shows $y \in \text{supp } N$
proof –
from $\text{Eq } \langle x \neq y \rangle \langle x \# \text{yvec} \rangle \langle y \# \text{yvec} \rangle$ **have** $(\nu * \text{xvec})M \prec' P = (\nu * \text{yvec})([(x, y)] \cdot N) \prec' ([x, y] \cdot Q)$
by(*simp add: boundOutput.inject alpha eqvts*)
then obtain p **where** $S: \text{set } p \subseteq \text{set } \text{xvec} \times \text{set } \text{yvec}$ **and** $M = p \cdot [(x, y)] \cdot N$
and *distinctPerm* p **using** $\langle \text{xvec} \# * \text{yvec} \rangle$
by(*blast dest: boundOutputChainEq'*)
with $\langle x \in \text{supp } M \rangle$ **have** $x \in \text{supp}(p \cdot [(x, y)] \cdot N)$ **by** *simp*
then have $(p \cdot x) \in p \cdot \text{supp}(p \cdot [(x, y)] \cdot N)$
by(*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# \text{xvec} \rangle \langle x \# \text{yvec} \rangle S \langle \text{distinctPerm } p \rangle$ **have** $x \in \text{supp}([(x, y)] \cdot N)$
by(*simp add: eqvts*)
then have $([(x, y)] \cdot x) \in ([x, y] \cdot (\text{supp}([(x, y)] \cdot N))$
by(*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \neq y \rangle$ **show** *?thesis* **by**(*simp add: calc-atm eqvts*)
qed

lemma *boundOutputChainOpenIH:*

fixes $\text{xvec} :: \text{name list}$
and $x :: \text{name}$
and $B :: ('a::\text{fs-name}, 'b::\text{fs-name}, 'c::\text{fs-name}) \text{boundOutput}$
and $\text{yvec} :: \text{name list}$
and $y :: \text{name}$
and $B' :: ('a, 'b, 'c) \text{boundOutput}$

assumes $\text{Eq}: (\nu * \text{xvec})(\nu x)B = (\nu * \text{yvec})(\nu y)B'$

and $L: \text{length } \text{xvec} = \text{length } \text{yvec}$
and $\text{xFresh}B': x \# B'$
and $\text{xFresh}\text{xvec}: x \# \text{xvec}$
and $\text{xFresh}\text{yvec}: x \# \text{yvec}$

shows $(\nu * \text{xvec})B = (\nu * \text{yvec})([(x, y)] \cdot B')$

using *assms*

proof(*induct n==length xvec arbitrary: xvec yvec y B' rule: nat.induct*)

case(*zero xvec yvec y B'*)

have $0 = \text{length } \text{xvec}$ **and** $\text{length } \text{xvec} = \text{length } \text{yvec}$ **by** *fact+*

moreover have $(\nu * \text{xvec})(\nu x)B = (\nu * \text{yvec})(\nu y)B'$ **by** *fact*

ultimately show *?case* **by**(*auto simp add: boundOutput.inject alpha*)

next

case(*Suc n xvec yvec y B'*)

have $L: \text{length } \text{xvec} = \text{length } \text{yvec}$ **and** $\text{Suc } n = \text{length } \text{xvec}$ **by** *fact+*

then obtain $x' \text{xvec}' y' \text{yvec}'$ **where** $\text{xEq}: \text{xvec} = x' \# \text{xvec}'$ **and** $\text{yEq}: \text{yvec} = y' \# \text{yvec}'$

and $L': \text{length } \text{xvec}' = \text{length } \text{yvec}'$

by(*cases xvec, auto, cases yvec, auto*)

have $\text{xFresh}B': x \# B'$ **by** *fact*

have $x \# xvec$ **and** $x \# yvec$ **by** *fact+*
with $xEq\ yEq$ **have** $xineqx': x \neq x'$ **and** $xFreshxvec': x \# xvec'$
and $xineqy': x \neq y'$ **and** $xFreshyvec': x \# yvec'$
by *simp+*
have $(\nu*xvec)(\nu x)B = (\nu*yvec)(\nu y)B'$ **by** *fact*
with $xEq\ yEq$ **have** $Eq: (\nu x')((\nu*xvec')(\nu x)B) = (\nu y')((\nu*yvec')(\nu y)B')$ **by**
simp
have $IH: \bigwedge xvec\ yvec\ y\ B'$.
 $\llbracket n = length\ xvec; (\nu*xvec)(\nu x)B = (\nu*yvec)(\nu y)B'; length\ xvec = length\ yvec; x \# B'; x \# xvec; x \# yvec \rrbracket$
 $\implies (\nu*xvec)B = (\nu*yvec)((x, y) \cdot B')$ **by** *fact*
have $Suc\ n = length\ xvec$ **by** *fact*
with xEq **have** $L'': n = length\ xvec'$ **by** *simp*
have $(\nu x')((\nu*xvec')B) = (\nu y')((\nu*yvec')((x, y) \cdot B'))$
proof (*cases* $x'=y'$)
assume $x'eqy': x' = y'$
with Eq **have** $(\nu*xvec')(\nu x)B = (\nu*yvec')(\nu y)B'$ **by** (*simp add: boundOutput.inject alpha*)
then **have** $(\nu*xvec')B = (\nu*yvec')((x, y) \cdot B')$ **using** $L'\ xFreshB'\ xFreshxvec'\ xFreshyvec'\ L''$ **by** (*metis IH*)
with $x'eqy'$ **show** *?thesis* **by** (*simp add: boundOutput.inject alpha*)
next
assume $x'ineqy': x' \neq y'$
with Eq **have** $Eq': (\nu*xvec')(\nu x)B = (\nu*((x', y') \cdot yvec'))(\nu((x', y') \cdot y))((x', y') \cdot B')$
and $x'FreshB': x' \# (\nu*yvec')(\nu y)B'$
by (*simp add: boundOutput.inject alpha eqvts*)
from L' **have** $length\ xvec' = length\ ((x', y') \cdot yvec')$ **by** *simp*
moreover **from** $xineqx'\ xineqy'\ xFreshB'$ **have** $x \# [(x', y') \cdot B']$ **by** (*simp add: fresh-left calc-atm*)
moreover **from** $xineqx'\ xineqy'\ xFreshyvec'$ **have** $x \# [(x', y') \cdot yvec']$ **by** (*simp add: fresh-left calc-atm*)
ultimately **have** $(\nu*xvec')B = (\nu*((x', y') \cdot yvec'))((x, ((x', y') \cdot y))) \cdot [(x', y') \cdot B']$ **using** $Eq'\ xFreshxvec'\ L''$
by (*metis IH*)
moreover **from** $x'FreshB'$ **have** $x' \# (\nu*yvec')((x, y) \cdot B')$
proof (*cases* $x' \# yvec'$)
assume $x' \# yvec'$
with $x'FreshB'$ **have** $x'FreshB': x' \# (\nu y)B'$
by (*simp add: fresh-def BOresChainSupp*)
show *?thesis*
proof (*cases* $x'=y$)
assume $x'eqy: x' = y$
show *?thesis*
proof (*cases* $x=y$)
assume $x=y$
with $xFreshB'\ x'eqy$ **show** *?thesis* **by** (*simp add: BOresChainSupp fresh-def*)
next
assume $x \neq y$

```

    with ⟨x ‡ B'⟩ have y ‡ [(x, y)] · B' by(simp add: fresh-left calc-atm)
    with x'eqy show ?thesis by(simp add: BOresChainSupp fresh-def)
  qed
next
  assume x'ineqy: x' ≠ y
  with x'FreshB' have x' ‡ B' by(simp add: abs-fresh)
  with xineqx' x'ineqy have x' ‡ [(x, y)] · B' by(simp add: fresh-left calc-atm)
  then show ?thesis by(simp add: BOresChainSupp fresh-def)
  qed
next
  assume ¬x' ‡ yvec'
  then show ?thesis by(simp add: BOresChainSupp fresh-def)
  qed
ultimately show ?thesis using x'ineqy' xineqx' xineqy'
  apply(simp add: boundOutput.inject alpha eqvts)
  apply(subst perm-compose[of [(x', y^)])
  by(simp add: calc-atm)
  qed
with xEq yEq show ?case by simp
qed

```

lemma *boundOutputPar1Dest*:

```

  fixes xvec :: name list
  and M     :: 'a::fs-name
  and P     :: ('a, 'b::fs-name, 'c::fs-name) psi
  and yvec  :: name list
  and N     :: 'a
  and Q     :: ('a, 'b, 'c) psi
  and R     :: ('a, 'b, 'c) psi

```

```

assumes (ν*xvec)M <' P = (ν*yvec)N <' (Q ‖ R)
  and   xvec ‡* R
  and   yvec ‡* R

```

obtains *T* where $P = T \parallel R$ and $(\nu*xvec)M <' T = (\nu*yvec)N <' Q$

proof –

```

  assume  $\bigwedge T. \llbracket P = T \parallel R; (\nu*xvec)M <' T = (\nu*yvec)N <' Q \rrbracket \implies$  thesis

```

```

  moreover obtain n where  $n = \text{length } xvec$  by auto

```

```

  with assms have  $\exists T. P = T \parallel R \wedge (\nu*xvec)M <' T = (\nu*yvec)N <' Q$ 

```

```

proof(induct n arbitrary: xvec yvec M N P Q R)

```

```

  case(0 xvec yvec M N P Q R)

```

```

  have Eq:  $(\nu*xvec)M <' P = (\nu*yvec)N <' (Q \parallel R)$  by fact

```

```

  from  $\langle 0 = \text{length } xvec \rangle$  have  $xvec = []$  by auto

```

```

  moreover with Eq have  $yvec = []$ 

```

```

    by(cases yvec) auto

```

```

  ultimately show ?case using Eq

```

```

    by(simp add: boundOutput.inject)

```

```

next

```

```

  case(Suc n xvec yvec M N P Q R)

```

```

from  $\langle \text{Suc } n = \text{length } xvec \rangle$ 
obtain  $x \ xvec'$  where  $xvec = x \# xvec'$  and  $\text{length } xvec' = n$ 
  by  $(\text{cases } xvec) \ \text{auto}$ 
from  $\langle (\nu * xvec) M \prec' P = (\nu * yvec) N \prec' (Q \parallel R) \rangle \langle xvec = x \# xvec' \rangle$ 
obtain  $y \ yvec'$  where  $(\nu * (x \# xvec')) M \prec' P = (\nu * (y \# yvec')) N \prec' (Q \parallel R)$ 
  and  $yvec = y \# yvec'$ 
  by  $(\text{cases } yvec) \ \text{auto}$ 
then have  $EQ: (\nu x)((\nu * xvec') M \prec' P) = (\nu y)((\nu * yvec') N \prec' (Q \parallel R))$ 
  by  $\text{simp}$ 
from  $\langle xvec \#* R \rangle \langle yvec \#* R \rangle \langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$ 
have  $x \# R$  and  $xvec' \#* R$  and  $y \# R$  and  $yvec' \#* R$  by  $\text{auto}$ 
  have  $IH: \bigwedge xvec \ yvec \ M \ N \ P \ Q \ R. \llbracket (\nu * xvec) M \prec' (P :: ('a, 'b, 'c) \ \text{psi}) =$ 
 $(\nu * yvec) N \prec' (Q \parallel R); xvec \#* R; yvec \#* R; n = \text{length } xvec \rrbracket \implies \exists T. P = T \parallel$ 
 $R \wedge (\nu * xvec) M \prec' T = (\nu * yvec) N \prec' Q$ 
  by  $\text{fact}$ 
show  $?case$ 
proof  $(\text{cases } x = y)$ 
  assume  $x = y$ 
  with  $EQ$  have  $(\nu * xvec') M \prec' P = (\nu * yvec') N \prec' (Q \parallel R)$ 
    by  $(\text{simp add: boundOutput.inject alpha})$ 
  with  $\langle xvec' \#* R \rangle \langle yvec' \#* R \rangle \langle \text{length } xvec' = n \rangle$ 
obtain  $T$  where  $P = T \parallel R$  and  $(\nu * xvec') M \prec' T = (\nu * yvec') N \prec' Q$ 
  by  $(\text{auto dest: IH})$ 
  with  $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle x = y \rangle$  show  $?case$ 
  by  $(\text{force simp add: boundOutput.inject alpha})$ 
next
  assume  $x \neq y$ 
  with  $EQ \langle x \# R \rangle \langle y \# R \rangle$ 
have  $(\nu * xvec') M \prec' P = (\nu * ((x, y) \cdot yvec')) ((x, y) \cdot N) \prec' (((x, y) \cdot Q)$ 
 $\parallel R)$ 
  and  $xFreshQR: x \# (\nu * yvec') N \prec' (Q \parallel R)$ 
  by  $(\text{simp add: boundOutput.inject alpha eqts})$ 
  moreover from  $\langle yvec' \#* R \rangle$  have  $((x, y) \cdot yvec') \#* ((x, y) \cdot R)$ 
  by  $(\text{simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]})$ 
  with  $\langle x \# R \rangle \langle y \# R \rangle$  have  $((x, y) \cdot yvec') \#* R$  by  $\text{simp}$ 
  moreover note  $\langle xvec' \#* R \rangle \langle \text{length } xvec' = n \rangle$ 
  ultimately obtain  $T$  where  $P = T \parallel R$  and  $A: (\nu * xvec') M \prec' T = (\nu * ((x,$ 
 $y) \cdot yvec')) ((x, y) \cdot N) \prec' ((x, y) \cdot Q)$ 
  by  $(\text{auto dest: IH})$ 

  from  $A$  have  $(\nu x)((\nu * xvec') M \prec' T) = (\nu x)((\nu * ((x, y) \cdot yvec')) ((x, y)$ 
 $\cdot N) \prec' ((x, y) \cdot Q))$ 
  by  $(\text{simp add: boundOutput.inject alpha})$ 
  moreover from  $xFreshQR$  have  $x \# (\nu * yvec') N \prec' Q$ 
  by  $(\text{force simp add: boundOutput.Fresh})$ 
  ultimately show  $?thesis$  using  $\langle P = T \parallel R \rangle \langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$ 
 $xFreshQR$ 
  by  $(\text{force simp add: alphaBoundOutput name-swap eqts})$ 
qed

```


qed
ultimately show *?thesis*
by *blast*
qed

lemma *boundOutputPar1Dest'*:

fixes *xvec* :: name list
and *M* :: 'a::fs-name
and *P* :: ('a, 'b::fs-name, 'c::fs-name) psi
and *yvec* :: name list
and *N* :: 'a
and *Q* :: ('a, 'b, 'c) psi
and *R* :: ('a, 'b, 'c) psi

assumes $(\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R)$
and $xvec \#* yvec$

obtains $T \ p$ where $set \ p \subseteq set \ xvec \times set \ yvec$ and $P = T \parallel (p \cdot R)$ and
 $(\nu*xvec)M \prec' T = (\nu*yvec)N \prec' Q$

proof -

assume $\bigwedge p \ T. \llbracket set \ p \subseteq set \ xvec \times set \ yvec; P = T \parallel (p \cdot R); (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' Q \rrbracket \implies thesis$

moreover obtain n where $n = length \ xvec$ by *auto*

with *assms* have $\exists p \ T. set \ p \subseteq set \ xvec \times set \ yvec \wedge P = T \parallel (p \cdot R) \wedge (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' Q$

proof (induct n arbitrary: *xvec yvec M N P Q R*)

case ($0 \ xvec \ yvec \ M \ N \ P \ Q \ R$)

have *Eq*: $(\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R)$ by *fact*

from $\langle 0 = length \ xvec \rangle$ have $xvec = []$ by *auto*

moreover with *Eq* have $yvec = []$

by (cases *yvec*) *auto*

ultimately show *?case* using *Eq*

by (simp add: *boundOutput.inject*)

next

case (*Suc n xvec yvec M N P Q R*)

from $\langle Suc \ n = length \ xvec \rangle$

obtain $x \ xvec'$ where $xvec = x \# xvec'$ and $length \ xvec' = n$

by (cases *xvec*) *auto*

from $\langle (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R) \rangle \langle xvec = x \# xvec' \rangle$

obtain $y \ yvec'$ where $(\nu*(x \# xvec'))M \prec' P = (\nu*(y \# yvec'))N \prec' (Q \parallel R)$

and $yvec = y \# yvec'$

by (cases *yvec*) *auto*

then have *Eq*: $(\nu x)((\nu*xvec')M \prec' P) = (\nu y)((\nu*yvec')N \prec' (Q \parallel R))$

by *simp*

from $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle xvec \#* yvec \rangle$ have $x \neq y$ and $x \# yvec'$ and $y \# xvec'$ and $xvec' \#* yvec'$

by *auto*

from *Eq* $\langle x \neq y \rangle$ have *Eq'*: $(\nu*xvec')M \prec' P = [(x, y)] \cdot (\nu*yvec')N \prec' (Q \parallel R)$

and $xFreshQR: x \# (\nu * yvec') N \prec' (Q \parallel R)$
by (*simp add: boundOutput.inject alpha*)
have $IH: \bigwedge xvec\ yvec\ M\ N\ P\ Q\ R. \llbracket (\nu * xvec) M \prec' (P :: ('a, 'b, 'c)\ psi) = (\nu * yvec) N \prec' (Q \parallel R); xvec \# * yvec; n = length\ xvec \rrbracket \implies \exists p\ T. set\ p \subseteq set\ xvec \times set\ yvec \wedge P = T \parallel (p \cdot R) \wedge (\nu * xvec) M \prec' T = (\nu * yvec) N \prec' Q$
by *fact*
show *?case*
proof (*cases* $x \# (\nu * xvec') M \prec' P$)
assume $x \# (\nu * xvec') M \prec' P$
with Eq **have** $yFreshQR: y \# (\nu * yvec') N \prec' (Q \parallel R)$
by (*rule boundOutputEqFresh*)
with Eq' $xFreshQR$ **have** $(\nu * xvec') M \prec' P = (\nu * yvec') N \prec' (Q \parallel R)$
by *simp*
with $\langle xvec' \# * yvec' \rangle \langle length\ xvec' = n \rangle$
obtain $p\ T$ **where** $S: set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = T \parallel (p \cdot R)$
and $A: (\nu * xvec') M \prec' T = (\nu * yvec') N \prec' Q$
by (*auto dest: IH*)
from $yFreshQR\ xFreshQR$ **have** $yFreshQ: y \# (\nu * yvec') N \prec' Q$ **and** $xFreshQ: x \# (\nu * yvec') N \prec' Q$
by (*force simp add: BOresChainSupp fresh-def boundOutput.supp psi.supp*)
then **have** $(\nu x)((\nu * yvec') N \prec' Q) = (\nu y)((\nu * yvec') N \prec' Q)$ **by** (*subst alphaBoundOutput simp*)
with A **have** $(\nu x)((\nu * xvec') M \prec' T) = (\nu y)((\nu * yvec') N \prec' Q)$ **by** *simp*
with $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle S \langle P = T \parallel (p \cdot R) \rangle$ **show** *?case*
by *auto*
next
assume $\neg(x \# (\nu * xvec') M \prec' P)$
then **have** $x \in supp((\nu * xvec') M \prec' P)$ **by** (*simp add: fresh-def*)
with Eq **have** $y \in supp((\nu * yvec') N \prec' (Q \parallel R))$
by (*rule boundOutputEqSupp*)
then **have** $y \# yvec'$ **by** (*simp add: BOresChainSupp fresh-def*)
with $Eq' \langle x \# yvec' \rangle$ **have** $(\nu * xvec') M \prec' P = (\nu * yvec')(([(x, y)] \cdot N) \prec' ([[(x, y)] \cdot Q] \parallel [(x, y)] \cdot R))$
by (*simp add: eqvts*)
moreover **note** $\langle xvec' \# * yvec' \rangle \langle length\ xvec' = n \rangle$
ultimately **obtain** $p\ T$ **where** $S: set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = T \parallel (p \cdot [(x, y)] \cdot R)$ **and** $A: (\nu * xvec') M \prec' T = (\nu * yvec')([[(x, y)] \cdot N] \prec' ([[(x, y)] \cdot Q] \cdot R))$
by (*auto dest: IH*)
from S **have** $set(p @ [(x, y)]) \subseteq set(x \# xvec') \times set(y \# yvec')$ **by** *auto*
moreover **from** $\langle P = T \parallel (p \cdot [(x, y)] \cdot R) \rangle$ **have** $P = T \parallel ((p @ [(x, y)]) \cdot R)$
by (*simp add: pt2[OF pt-name-inst]*)
moreover **from** $xFreshQR$ **have** $xFreshQ: x \# (\nu * yvec') N \prec' Q$
by (*force simp add: BOresChainSupp fresh-def boundOutput.supp psi.supp*)
with $\langle x \# yvec' \rangle \langle y \# yvec' \rangle \langle x \neq y \rangle$ **have** $y \# (\nu * yvec')([[(x, y)] \cdot N] \prec' ([[(x, y)] \cdot Q] \cdot R))$
by (*simp add: fresh-left calc-atm*)

```

    with ⟨x # yvec'⟩ ⟨y # yvec'⟩ have (νx)((ν*yvec')((x, y) · N) <' ((x, y) ·
Q)) = (νy)((ν*yvec')N <' Q)
    by(subst alphaBoundOutput) (assumption | simp add: eqvts)+
    with A have (νx)((ν*xvec')M <' T) = (νy)((ν*yvec')N <' Q) by simp
    ultimately show ?thesis using ⟨xvec=x#xvec'⟩ ⟨yvec=y#yvec'⟩
    by - (rule exI[where x=p@[x, y]], force)
  qed
qed
ultimately show ?thesis
  by blast
qed

```

lemma boundOutputPar2Dest:

```

fixes xvec :: name list
and M    :: 'a::fs-name
and P    :: ('a, 'b::fs-name, 'c::fs-name) psi
and yvec :: name list
and N    :: 'a
and Q    :: ('a, 'b, 'c) psi
and R    :: ('a, 'b, 'c) psi

```

```

assumes (ν*xvec)M <' P = (ν*yvec)N <' (Q || R)
and xvec #* Q
and yvec #* Q

```

obtains T where P = Q || T and (ν*xvec)M <' T = (ν*yvec)N <' R

proof -

assume $\bigwedge T. \llbracket P = Q || T; (\nu*xvec)M <' T = (\nu*yvec)N <' R \rrbracket \implies thesis$

moreover obtain n where n = length xvec by auto

with assms have $\exists T. P = Q || T \wedge (\nu*xvec)M <' T = (\nu*yvec)N <' R$

proof(induct n arbitrary: xvec yvec M N P Q R)

case(0 xvec yvec M N P Q R)

have Eq: $(\nu*xvec)M <' P = (\nu*yvec)N <' (Q || R)$ by fact

from ⟨0 = length xvec⟩ have xvec = [] by auto

moreover with Eq have yvec = []

by(cases yvec) auto

ultimately show ?case using Eq

by(simp add: boundOutput.inject)

next

case(Suc n xvec yvec M N P Q R)

from ⟨Suc n = length xvec⟩

obtain x xvec' where xvec = x#xvec' and length xvec' = n

by(cases xvec) auto

from $(\nu*xvec)M <' P = (\nu*yvec)N <' (Q || R)$, ⟨xvec = x # xvec'⟩

obtain y yvec' where $(\nu*(x\#xvec'))M <' P = (\nu*(y\#yvec'))N <' (Q || R)$

and yvec = y#yvec'

by(cases yvec) auto

then have EQ: $(\nu x)((\nu*xvec')M <' P) = (\nu y)((\nu*yvec')N <' (Q || R))$

by simp

```

from ⟨xvec #* Q⟩ ⟨yvec #* Q⟩ ⟨xvec = x#xvec'⟩ ⟨yvec = y#yvec'⟩
have x # Q and xvec' #* Q and y # Q and yvec' #* Q by auto
have IH:  $\bigwedge xvec\ yvec\ M\ N\ P\ Q\ R. \llbracket (\nu*xvec)M \prec' (P::('a, 'b, 'c)\ psi) =$ 
 $(\nu*yvec)N \prec' (Q \parallel R); xvec\ #*\ Q; yvec\ #*\ Q; n = \text{length}\ xvec \rrbracket \implies \exists T. P = Q \parallel$ 
 $T \wedge (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' R$ 
by fact
show ?case
proof(cases x = y)
  assume x = y
  with EQ have  $(\nu*xvec')M \prec' P = (\nu*yvec')N \prec' (Q \parallel R)$ 
    by(simp add: boundOutput.inject alpha)
  with ⟨xvec' #* Q⟩ ⟨yvec' #* Q⟩ ⟨length xvec' = n⟩
  obtain T where P = Q  $\parallel$  T and  $(\nu*xvec')M \prec' T = (\nu*yvec')N \prec' R$ 
    by(auto dest: IH)
  with ⟨xvec=x#xvec'⟩ ⟨yvec=y#yvec'⟩ ⟨x=y⟩ show ?case
    by(force simp add: boundOutput.inject alpha)
next
  assume x ≠ y
  with EQ ⟨x # Q⟩ ⟨y # Q⟩
  have  $(\nu*xvec')M \prec' P = (\nu*([(x, y)] \cdot yvec'))([[(x, y)] \cdot N] \prec' (Q \parallel ([[(x, y)]$ 
 $\cdot R]))$ 
    and xFreshQR: x #  $(\nu*yvec')N \prec' (Q \parallel R)$ 
    by(simp add: boundOutput.inject alpha eqts)+
  moreover from ⟨yvec' #* Q⟩ have  $([(x, y)] \cdot yvec')\ #*\ ([[(x, y)] \cdot Q]$ 
    by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨x # Q⟩ ⟨y # Q⟩ have  $([(x, y)] \cdot yvec')\ #*\ Q$  by simp
  moreover note ⟨xvec' #* Q⟩ ⟨length xvec' = n⟩
  ultimately obtain T where P = Q  $\parallel$  T and A:  $(\nu*xvec')M \prec' T = (\nu*([(x,$ 
 $y)] \cdot yvec'))([[(x, y)] \cdot N] \prec' ([[(x, y)] \cdot R])$ 
    by(auto dest: IH)

  from A have  $(\nu x)(\nu*xvec')M \prec' T = (\nu x)(\nu*([(x, y)] \cdot yvec'))([[(x, y)]$ 
 $\cdot N] \prec' ([[(x, y)] \cdot R]))$ 
    by(simp add: boundOutput.inject alpha)
  moreover from xFreshQR have x #  $(\nu*yvec')N \prec' R$ 
    by(force simp add: boundOutput.Fresh)
  ultimately show ?thesis using ⟨P = Q  $\parallel$  T⟩ ⟨xvec=x#xvec'⟩ ⟨yvec=y#yvec'⟩
xFreshQR
by(force simp add: alphaBoundOutput name-swap eqts)
qed
qed
ultimately show ?thesis
by blast
qed

lemma boundOutputPar2Dest':
fixes xvec :: name list
and M :: 'a::fs-name
and P :: ('a, 'b::fs-name, 'c::fs-name) psi

```

and $yvec :: name\ list$
and $N :: 'a$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $(\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R)$
and $xvec \#* yvec$

obtains $T\ p$ **where** $set\ p \subseteq set\ xvec \times set\ yvec$ **and** $P = (p \cdot Q) \parallel T$ **and**
 $(\nu*xvec)M \prec' T = (\nu*yvec)N \prec' R$

proof –

assume $\bigwedge p\ T. \llbracket set\ p \subseteq set\ xvec \times set\ yvec; P = (p \cdot Q) \parallel T; (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' R \rrbracket \implies thesis$

moreover obtain n **where** $n = length\ xvec$ **by** *auto*

with *assms* **have** $\exists p\ T. set\ p \subseteq set\ xvec \times set\ yvec \wedge P = (p \cdot Q) \parallel T \wedge (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' R$

proof (*induct* n *arbitrary*: $xvec\ yvec\ M\ N\ P\ Q\ R$)

case ($0\ xvec\ yvec\ M\ N\ P\ Q\ R$)

have $Eq: (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R)$ **by** *fact*

from $\langle 0 = length\ xvec \rangle$ **have** $xvec = []$ **by** *auto*

moreover with Eq **have** $yvec = []$

by (*cases* $yvec$) *auto*

ultimately show $?case$ **using** Eq

by (*simp* *add*: *boundOutput.inject*)

next

case ($Suc\ n\ xvec\ yvec\ M\ N\ P\ Q\ R$)

from $\langle Suc\ n = length\ xvec \rangle$

obtain $x\ xvec'$ **where** $xvec = x \# xvec'$ **and** $length\ xvec' = n$

by (*cases* $xvec$) *auto*

from $\langle (\nu*xvec)M \prec' P = (\nu*yvec)N \prec' (Q \parallel R) \rangle \langle xvec = x \# xvec' \rangle$

obtain $y\ yvec'$ **where** $(\nu*(x \# xvec'))M \prec' P = (\nu*(y \# yvec'))N \prec' (Q \parallel R)$
and $yvec = y \# yvec'$

by (*cases* $yvec$) *auto*

then have $Eq: (\nu x)((\nu*xvec')M \prec' P) = (\nu y)((\nu*yvec')N \prec' (Q \parallel R))$

by *simp*

from $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle xvec \#* yvec \rangle$ **have** $x \neq y$ **and** $x \# yvec'$ **and** $y \# xvec'$ **and** $xvec' \#* yvec'$

by *auto*

from $Eq \langle x \neq y \rangle$ **have** $Eq': (\nu*xvec')M \prec' P = [(x, y)] \cdot (\nu*yvec')N \prec' (Q \parallel R)$

and $xFreshQR: x \# (\nu*yvec')N \prec' (Q \parallel R)$

by (*simp* *add*: *boundOutput.inject* *alpha*) $+$

have $IH: \llbracket xvec\ yvec\ M\ N\ P\ Q\ R. \llbracket (\nu*xvec)M \prec' (P::('a, 'b, 'c)\ psi) = (\nu*yvec)N \prec' (Q \parallel R); xvec \#* yvec; n = length\ xvec \rrbracket \implies \exists p\ T. set\ p \subseteq set\ xvec \times set\ yvec \wedge P = (p \cdot Q) \parallel T \wedge (\nu*xvec)M \prec' T = (\nu*yvec)N \prec' R \rrbracket$

by *fact*

show $?case$

proof (*cases* $x \# (\nu*xvec')M \prec' P$)

assume $x \# (\nu*xvec')M \prec' P$

with Eq **have** $yFreshQR: y \# (\nu * yvec')N \prec' (Q \parallel R)$
by(*rule boundOutputEqFresh*)
with $Eq' xFreshQR$ **have** $(\nu * xvec')M \prec' P = (\nu * yvec')N \prec' (Q \parallel R)$
by *simp*
with $\langle xvec' \#* yvec' \rangle \langle length\ xvec' = n \rangle$
obtain $p\ T$ **where** $S: set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = (p \cdot Q) \parallel T$
and $A: (\nu * xvec')M \prec' T = (\nu * yvec')N \prec' R$
by(*auto dest: IH*)
from $yFreshQR\ xFreshQR$ **have** $yFreshR: y \# (\nu * yvec')N \prec' R$ **and** $xFreshQ:$
 $x \# (\nu * yvec')N \prec' R$
by(*force simp add: BOresChainSupp fresh-def boundOutput.supp psi.supp*)
then **have** $(\nu x)((\nu * yvec')N \prec' R) = (\nu y)((\nu * yvec')N \prec' R)$ **by** (*subst*
alphaBoundOutput) *simp*+
with A **have** $(\nu x)((\nu * xvec')M \prec' T) = (\nu y)((\nu * yvec')N \prec' R)$ **by** *simp*
with $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle\ S \langle P = (p \cdot Q) \parallel T \rangle$ **show** *?case*
by *auto*
next
assume $\neg(x \# (\nu * xvec')M \prec' P)$
then **have** $x \in supp((\nu * xvec')M \prec' P)$ **by**(*simp add: fresh-def*)
with Eq **have** $y \in supp((\nu * yvec')N \prec' (Q \parallel R))$
by(*rule boundOutputEqSupp*)
then **have** $y \# yvec'$ **by**(*simp add: BOresChainSupp fresh-def*)
with $Eq' \langle x \# yvec' \rangle$ **have** $(\nu * xvec')M \prec' P = (\nu * yvec')(([(x, y)] \cdot N) \prec'$
 $(([(x, y)] \cdot Q) \parallel ([[(x, y)] \cdot R]))$
by(*simp add: eqvts*)
moreover **note** $\langle xvec' \#* yvec' \rangle \langle length\ xvec' = n \rangle$
ultimately **obtain** $p\ T$ **where** $S: set\ p \subseteq set\ xvec' \times set\ yvec'$ **and** $P = (p$
 $\cdot [(x, y)] \cdot Q) \parallel T$ **and** $A: (\nu * xvec')M \prec' T = (\nu * yvec')([(x, y)] \cdot N) \prec' ([[(x, y)]$
 $\cdot R)$
by(*auto dest: IH*)

from S **have** $set(p@[[(x, y)]) \subseteq set(x \# xvec') \times set(y \# yvec')$ **by** *auto*
moreover **from** $\langle P = (p \cdot [(x, y)] \cdot Q) \parallel T \rangle$ **have** $P = ((p @ [(x, y)]) \cdot Q)$
 $\parallel T$
by(*simp add: pt2[OF pt-name-inst]*)
moreover **from** $xFreshQR$ **have** $xFreshR: x \# (\nu * yvec')N \prec' R$
by(*force simp add: BOresChainSupp fresh-def boundOutput.supp psi.supp*)
with $\langle x \# yvec' \rangle \langle y \# yvec' \rangle \langle x \neq y \rangle$ **have** $y \# (\nu * yvec')([(x, y)] \cdot N) \prec' ([[(x,$
 $y)] \cdot R)$
by(*simp add: fresh-left calc-atm*)
with $\langle x \# yvec' \rangle \langle y \# yvec' \rangle$ **have** $(\nu x)((\nu * yvec')([(x, y)] \cdot N) \prec' ([[(x, y)] \cdot$
 $R]) = (\nu y)((\nu * yvec')N \prec' R)$
by(*subst alphaBoundOutput*) (*assumption* | *simp add: eqvts*)
with A **have** $(\nu x)((\nu * xvec')M \prec' T) = (\nu y)((\nu * yvec')N \prec' R)$ **by** *simp*
ultimately **show** *?thesis* **using** $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$
by(*force intro!: exI[where $x = p@[[(x, y)]]$]*)
qed
qed
ultimately **show** *?thesis*

by *blast*
qed

lemma *boundOutputApp*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *B* :: ('a::fs-name, 'b::fs-name, 'c::fs-name) *boundOutput*

shows $(\nu^*(xvec@yvec))B = (\nu^*xvec)((\nu^*yvec)B)$
by(*induct xvec*) *auto*

lemma *openInjectAux*:
fixes *xvec1* :: *name list*
and *x* :: *name*
and *xvec2* :: *name list*
and *yvec* :: *name list*

assumes $length(xvec1@x\#xvec2) = length\ yvec$

shows $\exists yvec1\ y\ yvec2. yvec = yvec1@y\#yvec2 \wedge length\ xvec1 = length\ yvec1 \wedge length\ xvec2 = length\ yvec2$
apply(*rule exI*[**where** $x=take\ (length\ xvec1)\ yvec$])
apply(*rule exI*[**where** $x=yvec\ !\ length\ xvec1$])
apply(*rule exI*[**where** $x=drop\ (length\ xvec1+1)\ yvec$])
using *assms* **by**(*auto simp add: id-take-nth-drop*)

lemma *boundOutputOpenDest*:
fixes *yvec* :: *name list*
and *M* :: ('a::fs-name
and *P* :: ('a, 'b::fs-name, 'c::fs-name) *psi*
and *xvec1* :: *name list*
and *x* :: *name*
and *xvec2* :: *name list*
and *N* :: 'a
and *Q* :: ('a, 'b, 'c) *psi*

assumes *Eq*: $(\nu^*(xvec1@x\#xvec2))M \prec' P = (\nu^*yvec)N \prec' Q$
and $x \# xvec1$
and $x \# yvec$
and $x \# N$
and $x \# Q$
and *distinct yvec*

obtains *yvec1 y yvec2* **where** $yvec=yvec1@y\#yvec2$ **and** $length\ xvec1 = length\ yvec1$ **and** $length\ xvec2 = length\ yvec2$
and $(\nu^*(xvec1@xvec2))M \prec' P = (\nu^*(yvec1@yvec2))([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$
proof –

assume *Ass*: $\bigwedge yvec1\ y\ yvec2.$
 $\llbracket yvec = yvec1\ @\ y\ \#\ yvec2; length\ xvec1 = length\ yvec1; length\ xvec2 =$
 $length\ yvec2;$
 $(\nu*(xvec1\ @\ xvec2))M\ \prec' P = (\nu*(yvec1\ @\ yvec2))(\llbracket(x, y)\ \cdot\ N\ \prec' (\llbracket(x,$
 $y)\ \cdot\ Q\ \rrbracket)$
 $\implies thesis$
from *Eq* **have** $length(xvec1\ @x\ \#\ xvec2) = length\ yvec$ **by** (*rule boundOutputChainEqLength*)
then obtain *yvec1 y yvec2* **where** *A*: $yvec = yvec1\ @y\ \#\ yvec2$ **and** $length\ xvec1$
 $= length\ yvec1$
and $length\ xvec2 = length\ yvec2$
by (*metis openInjectAux sym*)

from $\langle distinct\ yvec \rangle A$ **have** $y\ \#\ yvec2$ **by** *simp*
from $A\ \langle x\ \#\ yvec \rangle$ **have** $x\ \#\ yvec2$ **and** $x\ \#\ yvec1$ **by** *simp+*
with *Eq* $\langle length\ xvec1 = length\ yvec1 \rangle\ \langle x\ \#\ N \rangle\ \langle x\ \#\ Q \rangle\ \langle y\ \#\ yvec2 \rangle\ \langle x\ \#\ xvec1 \rangle A$
have $(\nu*(xvec1\ @xvec2))M\ \prec' P = (\nu*(yvec1\ @yvec2))(\llbracket(x, y)\ \cdot\ N\ \prec' (\llbracket(x, y)$
 $\cdot\ Q\ \rrbracket)$
by (*force dest: boundOutputChainOpenIH simp add: boundOutputApp BOresChain-*
 $Supp\ fresh-def\ boundOutput.supp\ eqvts$)
with $\langle length\ xvec1 = length\ yvec1 \rangle\ \langle length\ xvec2 = length\ yvec2 \rangle A$ *Ass* **show**
 $?thesis$
by *blast*
qed

lemma *boundOutputOpenDest'*:

fixes *yvec* $:: name\ list$
and *M* $:: 'a::fs-name$
and *P* $:: ('a, 'b::fs-name, 'c::fs-name)\ psi$
and *xvec1* $:: name\ list$
and *x* $:: name$
and *xvec2* $:: name\ list$
and *N* $:: 'a$
and *Q* $:: ('a, 'b, 'c)\ psi$

assumes *Eq*: $(\nu*(xvec1\ @x\ \#\ xvec2))M\ \prec' P = (\nu*yvec)N\ \prec' Q$
and $x\ \#\ xvec1$
and $x\ \#\ yvec$
and $x\ \#\ N$
and $x\ \#\ Q$

obtains *yvec1 y yvec2* **where** $yvec = yvec1\ @y\ \#\ yvec2$ **and** $length\ xvec1 = length\ yvec1$
and $length\ xvec2 = length\ yvec2$

and $(\nu*(xvec1\ @xvec2))M\ \prec' P = (\nu*(yvec1\ @\llbracket(x, y)\ \cdot\ yvec2\ \rrbracket))(\llbracket(x, y)\ \cdot\ N\ \prec' (\llbracket(x, y)\ \cdot\ Q\ \rrbracket)$

proof –

assume *Ass*: $\bigwedge yvec1\ y\ yvec2.$

$\llbracket yvec = yvec1\ @\ y\ \#\ yvec2; length\ xvec1 = length\ yvec1; length\ xvec2 =$
 $length\ yvec2;$

$(\nu^*(xvec1 @ xvec2))M \prec' P = (\nu^*(yvec1 @ ((x, y) \cdot yvec2)))[(x, y) \cdot N] \prec'([(x, y)] \cdot Q]$
 $\implies thesis$
from *Eq* **have** $length(xvec1 @ x\#xvec2) = length yvec$ **by** (*rule boundOutputChainEqLength*)
then obtain *yvec1 yvec2* **where** *A*: $yvec = yvec1 @ y\#yvec2$ **and** $length xvec1 = length yvec1$
and $length xvec2 = length yvec2$
by (*metis openInjectAux sym*)

from *A* $\langle x \# yvec \rangle$ **have** $x \# yvec2$ **and** $x \# yvec1$ **by** *simp+*
with *Eq* $\langle length xvec1 = length yvec1 \rangle \langle x \# N \rangle \langle x \# Q \rangle \langle x \# xvec1 \rangle$ *A*
have $(\nu^*(xvec1 @ xvec2))M \prec' P = (\nu^*(yvec1 @ ((x, y) \cdot yvec2)))[(x, y) \cdot N] \prec'([(x, y)] \cdot Q]$
by (*force dest: boundOutputChainOpenIH simp add: boundOutputApp BOresChain-Supp fresh-def boundOutput.supp eqvts*)
with $\langle length xvec1 = length yvec1 \rangle \langle length xvec2 = length yvec2 \rangle$ *A Ass* **show** *?thesis*
by *blast*
qed

lemma *boundOutputScopeDest*:

fixes *xvec* :: *name list*
and *M* :: '*a*::*fs-name*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *yvec* :: *name list*
and *N* :: '*a*
and *x* :: *name*
and *Q* :: ('*a*, '*b*, '*c*) *psi*

assumes $(\nu^*xvec)M \prec' P = (\nu^*yvec)N \prec' (\nu z)Q$
and $z \# xvec$
and $z \# yvec$

obtains *R* **where** $P = (\nu z)R$ **and** $(\nu^*xvec)M \prec' R = (\nu^*yvec)N \prec' Q$

proof –

assume $\bigwedge R. \llbracket P = (\nu z)R; (\nu^*xvec)M \prec' R = (\nu^*yvec)N \prec' Q \rrbracket \implies thesis$

moreover obtain *n* **where** $n = length xvec$ **by** *auto*

with *assms* **have** $\exists R. P = (\nu z)R \wedge (\nu^*xvec)M \prec' R = (\nu^*yvec)N \prec' Q$

proof (*induct n arbitrary: xvec yvec M N P Q z*)

case (*0 xvec yvec M N P Q z*)

have *Eq*: $(\nu^*xvec)M \prec' P = (\nu^*yvec)N \prec' (\nu z)Q$ **by** *fact*

from $\langle 0 = length xvec \rangle$ **have** $xvec = []$ **by** *auto*

moreover with *Eq* **have** $yvec = []$

by (*cases yvec*) *auto*

ultimately show *?case* **using** *Eq*

by (*simp add: boundOutput.inject*)

next

case (*Suc n xvec yvec M N P Q z*)

from $\langle Suc n = length xvec \rangle$

```

obtain  $x \text{ } xvec'$  where  $xvec = x \# xvec'$  and  $length \text{ } xvec' = n$ 
  by(cases xvec) auto
from  $\langle (\nu * xvec) M \prec' P = (\nu * yvec) N \prec' (\nu z) Q \rangle \langle xvec = x \# xvec' \rangle$ 
obtain  $y \text{ } yvec'$  where  $(\nu * (x \# xvec')) M \prec' P = (\nu * (y \# yvec')) N \prec' (\nu z) Q$ 
  and  $yvec = y \# yvec'$ 
  by(cases yvec) auto
then have  $EQ: (\nu x)((\nu * xvec') M \prec' P) = (\nu y)((\nu * yvec') N \prec' (\nu z) Q)$ 
  by simp
from  $\langle z \# xvec \rangle \langle z \# yvec \rangle \langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$ 
have  $z \neq x$  and  $z \neq y$  and  $z \# xvec'$  and  $z \# yvec'$ 
  by simp+
  have  $IH: \bigwedge xvec \text{ } yvec \text{ } M \text{ } N \text{ } P \text{ } Q \text{ } z. \llbracket (\nu * xvec) M \prec' (P :: ('a, 'b, 'c) \text{ } psi) =$ 
 $(\nu * yvec) N \prec' (\nu z) Q; z \# xvec; z \# yvec; n = length \text{ } xvec \rrbracket \implies \exists R. P = (\nu z) R \wedge$ 
 $(\nu * xvec) M \prec' R = (\nu * yvec) N \prec' Q$ 
  by fact
show ?case
proof(cases x = y)
  assume  $x = y$ 
  with  $EQ$  have  $(\nu * xvec') M \prec' P = (\nu * yvec') N \prec' (\nu z) Q$ 
    by(simp add: boundOutput.inject alpha)
  with  $\langle z \# xvec' \rangle \langle z \# yvec' \rangle \langle length \text{ } xvec' = n \rangle$ 
obtain  $R$  where  $P = (\nu z) R$  and  $(\nu * xvec') M \prec' R = (\nu * yvec') N \prec' Q$ 
  by(auto dest: IH)
  with  $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle x = y \rangle$  show ?case
  by(force simp add: boundOutput.inject alpha)
next
  assume  $x \neq y$ 
  with  $EQ \langle z \neq x \rangle \langle z \neq y \rangle$ 
have  $(\nu * xvec') M \prec' P = (\nu * ((x, y) \cdot yvec')((x, y) \cdot N) \prec' (\nu z)((x, y) \cdot Q)$ 
  and  $xFreshzQ: x \# (\nu * yvec') N \prec' (\nu z) Q$ 
  by(simp add: boundOutput.inject alpha eqts)
  moreover from  $\langle z \neq x \rangle \langle z \neq y \rangle \langle z \# yvec' \rangle \langle x \neq y \rangle$  have  $z \# ((x, y) \cdot yvec')$ 
  by(simp add: fresh-left calc-atm)
  moreover note  $\langle z \# xvec' \rangle \langle length \text{ } xvec' = n \rangle$ 
  ultimately obtain  $R$  where  $P = (\nu z) R$  and  $A: (\nu * xvec') M \prec' R = (\nu * ((x, y) \cdot yvec')((x, y) \cdot N) \prec' ((x, y) \cdot Q)$ 
  by(auto dest: IH)

  from  $A$  have  $(\nu x)((\nu * xvec') M \prec' R) = (\nu x)((\nu * ((x, y) \cdot yvec')((x, y) \cdot N) \prec' ((x, y) \cdot Q))$ 
  by(simp add: boundOutput.inject alpha)
  moreover from  $xFreshzQ \langle z \neq x \rangle$  have  $x \# (\nu * yvec') N \prec' Q$ 
  by(simp add: boundOutputFresh abs-fresh)
  ultimately show ?thesis using  $\langle P = (\nu z) R \rangle \langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$ 
 $xFreshzQ$ 
  by(force simp add: alphaBoundOutput name-swap eqts)
qed
qed

```

ultimately show *?thesis*
 by *blast*
 qed

nominal-datatype ('a, 'b, 'c) *residual* =
 RIn 'a::fs-name 'a ('a, 'b::fs-name, 'c::fs-name) *psi*
 | RBrIn 'a::fs-name 'a ('a, 'b::fs-name, 'c::fs-name) *psi*
 | ROut 'a ('a, 'b, 'c) *boundOutput*
 | RBrOut 'a ('a, 'b, 'c) *boundOutput*
 | RTau ('a, 'b, 'c) *psi*

nominal-datatype 'a *action* = In 'a::fs-name 'a (-|-) [90, 90] 90
 | BrIn 'a::fs-name 'a (i-|-) [90, 90] 90
 | Out 'a::fs-name name list 'a (-|ν*-|) (-) [90, 90, 90] 90
 | BrOut 'a::fs-name name list 'a (i-|ν*-|) (-) [90, 90, 90] 90
 | Tau (τ 90)

nominal-primrec *bn* :: ('a::fs-name) *action* ⇒ name list

where

bn (M(|N|)) = []
 | *bn* (iM(|N|)) = []
 | *bn* (M(|ν*xvec|)⟨N⟩) = *xvec*
 | *bn* (iM(|ν*xvec|)⟨N⟩) = *xvec*
 | *bn* (τ) = []

by(rule TrueI)+

lemma *bnEqvt[eqvt]*:

fixes *p* :: name prm
 and *α* :: ('a::fs-name) *action*

shows (*p* · *bn* *α*) = *bn*(*p* · *α*)

by(nominal-induct *α* rule: *action.strong-induct*) auto

nominal-primrec *create-residual* :: ('a::fs-name) *action* ⇒ ('a, 'b::fs-name, 'c::fs-name)
psi ⇒ ('a, 'b, 'c) *residual* (- < - [80, 80] 80)

where

(M(|N|)) < P = RIn M N P
 | (iM(|N|)) < P = RBrIn M N P
 | M(|ν*xvec|)⟨N⟩ < P = ROut M (|ν*xvec|)(N <' P)
 | (iM(|ν*xvec|)⟨N⟩) < P = RBrOut M (|ν*xvec|)(N <' P)
 | τ < P = (RTau P)

by(rule TrueI)+

nominal-primrec *subject* :: ('a::fs-name) *action* ⇒ 'a option

where

subject (M(|N|)) = Some M
 | *subject* (iM(|N|)) = Some M
 | *subject* (M(|ν*xvec|)⟨N⟩) = Some M
 | *subject* (iM(|ν*xvec|)⟨N⟩) = Some M

| *subject* (τ) = *None*
by(*rule TrueI*)+

nominal-primrec *object* :: ('a::fs-name) *action* \Rightarrow 'a *option*

where
object ($M(N)$) = *Some N*
| *object* (${}_iM(N)$) = *Some N*
| *object* ($M(\nu*xvec)(N)$) = *Some N*
| *object* (${}_iM(\nu*xvec)(N)$) = *Some N*
| *object* (τ) = *None*
by(*rule TrueI*)+

lemma *optionFreshChain[simp]*:

fixes *xvec* :: *name list*
and *X* :: *name set*

shows $xvec \#* (Some\ x) = xvec \#* x$
and $X \#* (Some\ x) = X \#* x$
and $xvec \#* None$
and $X \#* None$
by(*auto simp add: fresh-star-def fresh-some fresh-none*)

lemmas [*simp*] = *fresh-some fresh-none*

lemma *actionFresh[simp]*:

fixes *x* :: *name*
and α :: ('a::fs-name) *action*

shows $(x \# \alpha) = (x \# (subject\ \alpha) \wedge x \# (bn\ \alpha) \wedge x \# (object\ \alpha))$
by(*nominal-induct* α *rule: action.strong-induct*) *auto*

lemma *actionFreshChain[simp]*:

fixes *X* :: *name set*
and α :: ('a::fs-name) *action*
and *xvec* :: *name list*

shows $(X \#* \alpha) = (X \#* (subject\ \alpha) \wedge X \#* (bn\ \alpha) \wedge X \#* (object\ \alpha))$
and $(xvec \#* \alpha) = (xvec \#* (subject\ \alpha) \wedge xvec \#* (bn\ \alpha) \wedge xvec \#* (object\ \alpha))$
by(*auto simp add: fresh-star-def*)

lemma *subjectEqvt[eqvt]*:

fixes *p* :: *name prm*
and α :: ('a::fs-name) *action*

shows $(p \cdot subject\ \alpha) = subject(p \cdot \alpha)$
by(*nominal-induct* α *rule: action.strong-induct*) *auto*

lemma *okjectEqvt[eqvt]*:

fixes *p* :: *name prm*

```

and  $\alpha :: ('a::fs\text{-name})\ \text{action}$ 

shows  $(p \cdot \text{object } \alpha) = \text{object}(p \cdot \alpha)$ 
by(nominal-induct  $\alpha$  rule: action.strong-induct) auto

lemma create-residualEqvt[eqvt]:
fixes  $p :: \text{name } prm$ 
and  $\alpha :: ('a::fs\text{-name})\ \text{action}$ 
and  $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$ 

shows  $(p \cdot (\alpha \prec P)) = (p \cdot \alpha) \prec (p \cdot P)$ 
by(nominal-induct  $\alpha$  rule: action.strong-induct)
(auto simp add: eqvts)

lemma residualFresh:
fixes  $x :: \text{name}$ 
and  $\alpha :: ('a::fs\text{-name})\ \text{action}$ 
and  $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$ 

shows  $(x \# (\alpha \prec P)) = (x \# (\text{subject } \alpha \wedge (x \in (\text{set}(\text{bn}(\alpha)))) \vee (x \# \text{object}(\alpha) \wedge x \# P)))$ 
by(nominal-induct  $\alpha$  rule: action.strong-induct)
(auto simp add: fresh-some fresh-none boundOutputFresh)

lemma residualFresh2[simp]:
fixes  $x :: \text{name}$ 
and  $\alpha :: ('a::fs\text{-name})\ \text{action}$ 
and  $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$ 

assumes  $x \# \alpha$ 
and  $x \# P$ 

shows  $x \# \alpha \prec P$ 
using assms
by(nominal-induct  $\alpha$  rule: action.strong-induct) auto

lemma residualFreshChain2[simp]:
fixes  $xvec :: \text{name list}$ 
and  $X :: \text{name set}$ 
and  $\alpha :: ('a::fs\text{-name})\ \text{action}$ 
and  $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$ 

shows  $\llbracket xvec \#* \alpha; xvec \#* P \rrbracket \Longrightarrow xvec \#* (\alpha \prec P)$ 
and  $\llbracket X \#* \alpha; X \#* P \rrbracket \Longrightarrow X \#* (\alpha \prec P)$ 
by(auto simp add: fresh-star-def)

lemma residualFreshSimp[simp]:
fixes  $x :: \text{name}$ 
and  $M :: ('a::fs\text{-name})$ 

```

and $N :: 'a$
and $P :: ('a, 'b::fs-name, 'c::fs-name) psi$

shows $x \# (M(N) \prec P) = (x \# M \wedge x \# N \wedge x \# P)$
and $x \# (i M(N) \prec P) = (x \# M \wedge x \# N \wedge x \# P)$
and $x \# (M(\nu*xvec)\langle N \rangle \prec P) = (x \# M \wedge x \# (\nu*xvec)\langle N \rangle \prec' P)$
and $x \# (iM(\nu*xvec)\langle N \rangle \prec P) = (x \# M \wedge x \# (\nu*xvec)\langle N \rangle \prec' P)$
and $x \# (\tau \prec P) = (x \# P)$
by(*auto simp add: residualFresh*)

lemma *residualInject'*:

shows $(\alpha \prec P = RIn M N Q) = (P = Q \wedge \alpha = M(N))$
and $(\alpha \prec P = RBrIn M N Q) = (P = Q \wedge \alpha = iM(N))$
and $(\alpha \prec P = ROut M B) = (\exists xvec N. \alpha = M(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
and $(\alpha \prec P = RBrOut M B) = (\exists xvec N. \alpha = iM(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
and $(\alpha \prec P = RTau Q) = (\alpha = \tau \wedge P = Q)$
and $(RIn M N Q = \alpha \prec P) = (P = Q \wedge \alpha = M(N))$
and $(RBrIn M N Q = \alpha \prec P) = (P = Q \wedge \alpha = iM(N))$
and $(ROut M B = \alpha \prec P) = (\exists xvec N. \alpha = M(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
and $(RBrOut M B = \alpha \prec P) = (\exists xvec N. \alpha = iM(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
and $(RTau Q = \alpha \prec P) = (\alpha = \tau \wedge P = Q)$
proof –
show $(\alpha \prec P = RIn M N Q) = (P = Q \wedge \alpha = M(N))$
by(*nominal-induct* α *rule: action.strong-induct*)
(auto simp add: residual.inject action.inject)
next
show $(\alpha \prec P = RBrIn M N Q) = (P = Q \wedge \alpha = iM(N))$
by(*nominal-induct* α *rule: action.strong-induct*)
(auto simp add: residual.inject action.inject)
next
show $(\alpha \prec P = ROut M B) = (\exists xvec N. \alpha = M(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
by(*nominal-induct* α *rule: action.strong-induct*)
(auto simp add: residual.inject action.inject)
next
show $(\alpha \prec P = RBrOut M B) = (\exists xvec N. \alpha = iM(\nu*xvec)\langle N \rangle \wedge B = (\nu*xvec)\langle N \rangle \prec' P)$
by(*nominal-induct* α *rule: action.strong-induct*)
(auto simp add: residual.inject action.inject)
next
show $(\alpha \prec P = RTau Q) = (\alpha = \tau \wedge P = Q)$
by(*nominal-induct* α *rule: action.strong-induct*)
(auto simp add: residual.inject action.inject)

```

next
  show ( $RIn\ M\ N\ Q = \alpha \prec P$ ) = ( $P = Q \wedge \alpha = M(\downarrow N)$ )
    by(nominal-induct  $\alpha$  rule: action.strong-induct)
      (auto simp add: residual.inject action.inject)
next
  show ( $RBrIn\ M\ N\ Q = \alpha \prec P$ ) = ( $P = Q \wedge \alpha = \downarrow M(\downarrow N)$ )
    by(nominal-induct  $\alpha$  rule: action.strong-induct)
      (auto simp add: residual.inject action.inject)
next
  show ( $ROut\ M\ B = \alpha \prec P$ ) = ( $(\exists\ xvec\ N. \alpha = M(\downarrow \nu * xvec)\langle N \rangle \wedge B = (\downarrow \nu * xvec)\langle N \rangle \prec' P)$ )
    by(nominal-induct  $\alpha$  rule: action.strong-induct)
      (auto simp add: residual.inject action.inject)
next
  show ( $RBrOut\ M\ B = \alpha \prec P$ ) = ( $(\exists\ xvec\ N. \alpha = \downarrow M(\downarrow \nu * xvec)\langle N \rangle \wedge B = (\downarrow \nu * xvec)\langle N \rangle \prec' P)$ )
    by(nominal-induct  $\alpha$  rule: action.strong-induct)
      (auto simp add: residual.inject action.inject)
next
  show ( $RTau\ Q = \alpha \prec P$ ) = ( $\alpha = \tau \wedge P = Q$ )
    by(nominal-induct  $\alpha$  rule: action.strong-induct)
      (auto simp add: residual.inject action.inject)
qed

```

lemma *residualFreshChainSimp[simp]*:

```

fixes  $xvec :: name\ list$ 
  and  $X :: name\ set$ 
  and  $M :: 'a::fs-name$ 
  and  $N :: 'a$ 
  and  $yvec :: name\ list$ 
  and  $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$ 

```

```

shows  $xvec \#* (M(\downarrow N) \prec P) = (xvec \#* M \wedge xvec \#* N \wedge xvec \#* P)$ 
and  $xvec \#* (\downarrow M(\downarrow N) \prec P) = (xvec \#* M \wedge xvec \#* N \wedge xvec \#* P)$ 
and  $xvec \#* (M(\downarrow \nu * yvec)\langle N \rangle \prec P) = (xvec \#* M \wedge xvec \#* ((\downarrow \nu * yvec)\langle N \rangle \prec' P))$ 
and  $xvec \#* (\downarrow M(\downarrow \nu * yvec)\langle N \rangle \prec P) = (xvec \#* M \wedge xvec \#* ((\downarrow \nu * yvec)\langle N \rangle \prec' P))$ 
and  $xvec \#* (\tau \prec P) = (xvec \#* P)$ 
and  $X \#* (M(\downarrow N) \prec P) = (X \#* M \wedge X \#* N \wedge X \#* P)$ 
and  $X \#* (\downarrow M(\downarrow N) \prec P) = (X \#* M \wedge X \#* N \wedge X \#* P)$ 
and  $X \#* (M(\downarrow \nu * yvec)\langle N \rangle \prec P) = (X \#* M \wedge X \#* ((\downarrow \nu * yvec)\langle N \rangle \prec' P))$ 
and  $X \#* (\downarrow M(\downarrow \nu * yvec)\langle N \rangle \prec P) = (X \#* M \wedge X \#* ((\downarrow \nu * yvec)\langle N \rangle \prec' P))$ 
and  $X \#* (\tau \prec P) = (X \#* P)$ 
by(auto simp add: fresh-star-def)

```

lemma *residualFreshChainSimp2[simp]*:

```

fixes  $xvec :: name\ list$ 
  and  $X :: name\ set$ 
  and  $M :: 'a::fs-name$ 
  and  $N :: 'a$ 

```

and $yvec :: name\ list$
and $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$

shows $xvec \#^* (RIn\ M\ N\ P) = (xvec \#^* M \wedge xvec \#^* N \wedge xvec \#^* P)$
and $xvec \#^* (RBrIn\ M\ N\ P) = (xvec \#^* M \wedge xvec \#^* N \wedge xvec \#^* P)$
and $xvec \#^* (ROut\ M\ B) = (xvec \#^* M \wedge xvec \#^* B)$
and $xvec \#^* (RBrOut\ M\ B) = (xvec \#^* M \wedge xvec \#^* B)$
and $xvec \#^* (RTau\ P) = (xvec \#^* P)$
and $X \#^* (RIn\ M\ N\ P) = (X \#^* M \wedge X \#^* N \wedge X \#^* P)$
and $X \#^* (RBrIn\ M\ N\ P) = (X \#^* M \wedge X \#^* N \wedge X \#^* P)$
and $X \#^* (ROut\ M\ B) = (X \#^* M \wedge X \#^* B)$
and $X \#^* (RBrOut\ M\ B) = (X \#^* M \wedge X \#^* B)$
and $X \#^* (RTau\ P) = (X \#^* P)$
by(*auto simp add: fresh-star-def*)

lemma *freshResidual3[dest]*:
fixes $x :: name$
and $\alpha :: ('a::fs-name)\ action$
and $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$

assumes $x \# bn\ \alpha$
and $x \# \alpha \prec P$

shows $x \# \alpha$ **and** $x \# P$
using *assms*
by(*nominal-induct rule: action.strong-induct*) *auto*

lemma *freshResidualChain3[dest]*:
fixes $xvec :: name\ list$
and $\alpha :: ('a::fs-name)\ action$
and $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$

assumes $xvec \#^* (\alpha \prec P)$
and $xvec \#^* bn\ \alpha$

shows $xvec \#^* \alpha$ **and** $xvec \#^* P$
using *assms*
by(*nominal-induct rule: action.strong-induct*) *auto*

lemma *freshResidual4[dest]*:
fixes $x :: name$
and $\alpha :: ('a::fs-name)\ action$
and $P :: ('a, 'b::fs-name, 'c::fs-name)\ psi$

assumes $x \# \alpha \prec P$

shows $x \# subject\ \alpha$
using *assms*
by(*nominal-induct rule: action.strong-induct*) *auto*


```

lemma freshResidualChain4[dest]:
  fixes xvec :: name list
    and  $\alpha$  :: ('a::fs-name) action
    and  $P$  :: ('a, 'b::fs-name, 'c::fs-name) psi

assumes  $xvec \#* (\alpha \prec P)$ 

shows  $xvec \#* \text{subject } \alpha$ 
  using assms
  by(nominal-induct rule: action.strong-induct) auto

lemma alphaOutputResidual:
  fixes  $M$  :: 'a::fs-name
    and xvec :: name list
    and  $N$  :: 'a
    and  $P$  :: ('a, 'b::fs-name, 'c::fs-name) psi
    and  $p$  :: name prm

assumes  $(p \cdot xvec) \#* N$ 
  and  $(p \cdot xvec) \#* P$ 
  and  $\text{set } p \subseteq \text{set } xvec \times \text{set}(p \cdot xvec)$ 
  and  $\text{set } xvec \subseteq \text{set } yvec$ 

shows  $M(\nu^* yvec)\langle N \rangle \prec P = M(\nu^*(p \cdot yvec))\langle (p \cdot N) \rangle \prec (p \cdot P)$ 
  and  $\text{i}M(\nu^* yvec)\langle N \rangle \prec P = \text{i}M(\nu^*(p \cdot yvec))\langle (p \cdot N) \rangle \prec (p \cdot P)$ 
  using assms
  by(simp add: boundOutputChainAlpha'')+

lemmas[simp del] = create-residual.simps

lemma residualInject'':

assumes  $bn \alpha = bn \beta$ 

shows  $(\alpha \prec P = \beta \prec Q) = (\alpha = \beta \wedge P = Q)$ 
  using assms
  by(nominal-induct  $\alpha$  rule: action.strong-induct)
  (force simp add: residual.inject create-residual.simps residualInject' action.inject boundOutput.inject)+

lemmas residualInject = residual.inject create-residual.simps residualInject' residualInject''

lemma bnFreshResidual[simp]:
  fixes  $\alpha$  :: ('a::fs-name) action

shows  $(bn \alpha) \#* (\alpha \prec P) = bn \alpha \#* (\text{subject } \alpha)$ 
  by(nominal-induct  $\alpha$  rule: action.strong-induct)

```

(*auto simp add: residualFresh fresh-some fresh-star-def*)

lemma *actionCases*[*case-names cInput cBrInput cOutput cBrOutput cTau*]:
fixes $\alpha :: ('a::fs\text{-name})\ \text{action}$

assumes $\bigwedge M\ N. \alpha = M(\langle N \rangle) \implies \text{Prop}$
and $\bigwedge M\ N. \alpha = \iota M(\langle N \rangle) \implies \text{Prop}$
and $\bigwedge M\ \text{vec}\ N. \alpha = M(\langle \nu * \text{vec} \rangle \langle N \rangle) \implies \text{Prop}$
and $\bigwedge M\ \text{vec}\ N. \alpha = \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \implies \text{Prop}$
and $\alpha = \tau \implies \text{Prop}$

shows *Prop*
using *assms*
by(*nominal-induct* α *rule: action.strong-induct*) *auto*

lemma *actionPar1Dest*:
fixes $\alpha :: ('a::fs\text{-name})\ \text{action}$
and $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$
and $\beta :: ('a::fs\text{-name})\ \text{action}$
and $Q :: ('a, 'b, 'c)\ \text{psi}$
and $R :: ('a, 'b, 'c)\ \text{psi}$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $\text{bn}\ \alpha \# * \text{bn}\ \beta$

obtains $T\ p$ **where** $\text{set}\ p \subseteq \text{set}(\text{bn}\ \alpha) \times \text{set}(\text{bn}\ \beta)$ **and** $P = T \parallel (p \cdot R)$ **and** $\alpha \prec T = \beta \prec Q$
using *assms*
by(*cases rule: actionCases*[**where** $\alpha = \alpha$])
(*force simp add: residualInject dest: boundOutputPar1Dest*)**+**

lemma *actionPar2Dest*:
fixes $\alpha :: ('a::fs\text{-name})\ \text{action}$
and $P :: ('a, 'b::fs\text{-name}, 'c::fs\text{-name})\ \text{psi}$
and $\beta :: ('a::fs\text{-name})\ \text{action}$
and $Q :: ('a, 'b, 'c)\ \text{psi}$
and $R :: ('a, 'b, 'c)\ \text{psi}$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $\text{bn}\ \alpha \# * \text{bn}\ \beta$

obtains $T\ p$ **where** $\text{set}\ p \subseteq \text{set}(\text{bn}\ \alpha) \times \text{set}(\text{bn}\ \beta)$ **and** $P = (p \cdot Q) \parallel T$ **and** $\alpha \prec T = \beta \prec R$
using *assms*
by(*cases rule: actionCases*[**where** $\alpha = \alpha$])
(*force simp add: simp add: residualInject dest: boundOutputPar2Dest*)**+**

lemma *actionScopeDest*:
fixes $\alpha :: ('a::fs\text{-name})\ \text{action}$

```

    and P :: ('a, 'b::fs-name, 'c::fs-name) psi
  fixes β :: ('a::fs-name) action
    and x :: name
    and Q :: ('a, 'b, 'c) psi

assumes α < P = β < (νx)Q
  and x # bn α
  and x # bn β

obtains R where P = (νx)R and α < R = β < Q
  using assms boundOutputScopeDest
  by(cases rule: actionCases[where α=α]) (force simp add: residualInject)+

lemma emptyFreshName:
  fixes x :: name
    and M :: 'a::fs-name

assumes supp M = ({}::name set)

shows x # M
  using assms
  by(auto simp add: fresh-def)

lemma emptyFresh:
  fixes xvec :: name list
    and M :: 'a::fs-name

assumes supp M = ({}::name set)

shows xvec #* M
  using assms by (induct xvec, auto simp add: emptyFreshName)

lemma permEmptyEq:
  fixes p :: name prm
    and M :: 'a::fs-name

assumes suppE: supp M = ({}::name set)

shows (p · M) = M
proof(induct p)
  case Nil
  then show ?case by simp
next
  case(Cons a p)
  have p · M = M by(rule Cons)
  then have ([a] · p · M) = [a] · M by simp
  then have ((a#p) · M) = [a] · M
  by(simp add: pt2[OF pt-name-inst, symmetric])
  then show ?case using suppE perm-fresh-fresh

```

by(*cases a*) (*simp add: fresh-def*)
qed

abbreviation

outputJudge (*-(-)* [*110, 110*] *110*) **where** $M\langle N \rangle \equiv M(\nu^*(\square))\langle N \rangle$

abbreviation

brOutputJudge (*i-(-)* [*110, 110*] *110*) **where** $iM\langle N \rangle \equiv iM(\nu^*(\square))\langle N \rangle$

declare [[*unify-trace-bound=100*]]

locale *env* = *substPsi substTerm substAssert substCond* +
assertion SCompose' SImp' SBottom' SChanEq' SOutCon' SInCon'
for *substTerm* :: ('*a*::*fs-name*) \Rightarrow *name list* \Rightarrow '*a*::*fs-name list* \Rightarrow '*a*
and *substAssert* :: ('*b*::*fs-name*) \Rightarrow *name list* \Rightarrow '*a*::*fs-name list* \Rightarrow '*b*
and *substCond* :: ('*c*::*fs-name*) \Rightarrow *name list* \Rightarrow '*a*::*fs-name list* \Rightarrow '*c*
and *SCompose'* :: '*b* \Rightarrow '*b* \Rightarrow '*b*
and *SImp'* :: '*b* \Rightarrow '*c* \Rightarrow *bool*
and *SBottom'* :: '*b*
and *SChanEq'* :: '*a* \Rightarrow '*a* \Rightarrow '*c*
and *SOutCon'* :: '*a* \Rightarrow '*a* \Rightarrow '*c*
and *SInCon'* :: '*a* \Rightarrow '*a* \Rightarrow '*c*

begin

notation *SCompose'* (**infixr** \otimes *90*)

notation *SImp'* (*- \vdash -* [*85, 85*] *85*)

notation *FrameImp* (*- \vdash_F -* [*85, 85*] *85*)

abbreviation

FBottomJudge (\perp_F *90*) **where** $\perp_F \equiv (FAssert SBottom')$

notation *SChanEq'* (*- \leftrightarrow -* [*90, 90*] *90*)

notation *SOutCon'* (*- \preceq -* [*90, 90*] *90*)

notation *SInCon'* (*- \succeq -* [*90, 90*] *90*)

notation *substTerm* (*-[$::$]=*) [*100, 100, 100*] *100*)

notation *subs* (*-[$::$]=*) [*100, 100, 100*] *100*)

notation *AssertionStatEq* (*- \simeq -* [*80, 80*] *80*)

notation *FrameStatEq* (*- \simeq_F -* [*80, 80*] *80*)

notation *SBottom'* (**1** *190*)

abbreviation *insertAssertion'* (*insertAssertion*) **where** *insertAssertion'* \equiv *assertionAux.insertAssertion* (\otimes)

inductive *semantics* :: '*b* \Rightarrow ('*a*, '*b*, '*c*) *psi* \Rightarrow ('*a*, '*b*, '*c*) *residual* \Rightarrow *bool*
(*- \triangleright - \mapsto -* [*50, 50, 50*] *50*)

where

cInput: $\llbracket \Psi \vdash M \leftrightarrow K; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N; xvec \#* Tvec;$

length *xvec* = *length* *Tvec*;

$xvec \#* \Psi; xvec \#* M; xvec \#* K \rrbracket \Longrightarrow \Psi \triangleright M(\lambda * xvec N).P \mapsto$

$K(\llbracket (N[xvec::=Tvec]) \rrbracket) \prec P[xvec::=Tvec]$

| *cBrInput*: $\llbracket \Psi \vdash K \succeq M; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N; xvec \#* Tvec;$

length *xvec* = *length* *Tvec*;

$xvec \#* \Psi; xvec \#* M; xvec \#* K \rrbracket \Longrightarrow \Psi \triangleright M(\lambda * xvec N).P \mapsto$

$iK(\langle N[xvec::=Tvec] \rangle) \prec P[xvec::=Tvec]$
| *Output*: $\llbracket \Psi \vdash M \leftrightarrow K \rrbracket \Longrightarrow \Psi \triangleright M\langle N \rangle.P \mapsto K\langle N \rangle \prec P$
| *BrOutput*: $\llbracket \Psi \vdash M \preceq K \rrbracket \Longrightarrow \Psi \triangleright M\langle N \rangle.P \mapsto iK\langle N \rangle \prec P$
| *Case*: $\llbracket \Psi \triangleright P \mapsto Rs; (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P \rrbracket \Longrightarrow \Psi \triangleright \text{Cases } Cs \mapsto Rs$
| *cPar1*: $\llbracket (\Psi \otimes \Psi_Q) \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* P'; \text{distinct}(bn \alpha);$
 $bn \alpha \#* \Psi; bn \alpha \#* \Psi_Q; bn \alpha \#* Q; bn \alpha \#* P; bn \alpha \#* (\text{subject } \alpha) \rrbracket \Longrightarrow$
 $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P' \parallel Q)$
| *cPar2*: $\llbracket (\Psi \otimes \Psi_P) \triangleright Q \mapsto \alpha \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* Q'; \text{distinct}(bn \alpha);$
 $bn \alpha \#* \Psi; bn \alpha \#* \Psi_P; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* (\text{subject } \alpha) \rrbracket \Longrightarrow$
 $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$
| *cComm1*: $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
 $\text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu^*xvec)\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
 $\text{distinct } A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* K; A_Q \#* Q'; A_Q \#* xvec; \text{distinct } xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M;$
 $xvec \#* Q; xvec \#* K \rrbracket \Longrightarrow$
 $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*xvec)(P' \parallel Q')$
| *cComm2*: $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct}$
 $A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* K; A_Q \#* Q'; A_Q \#* xvec; \text{distinct } xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M;$
 $xvec \#* Q; xvec \#* K \rrbracket \Longrightarrow$
 $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*xvec)(P' \parallel Q')$
| *cBrMerge*: $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
 $\text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct}$
 $A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q;$
 $A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q' \rrbracket \Longrightarrow$
 $\Psi \triangleright P \parallel Q \mapsto iM\langle N \rangle \prec (P' \parallel Q')$
| *cBrComm1*: $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$

$distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec;$
 $A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_Q \#* xvec; distinct xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; xvec \#* M \implies$
 $\Psi \triangleright P \parallel Q \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec (P' \parallel Q')$
 $| cBrComm2: \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct$
 $A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec;$
 $A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_Q \#* xvec; distinct xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; xvec \#* M \implies$
 $\Psi \triangleright P \parallel Q \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec (P' \parallel Q')$
 $| cBrClose: \llbracket \Psi \triangleright P \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec P';$
 $x \in supp M;$
 $distinct xvec; xvec \#* \Psi; xvec \#* P;$
 $xvec \#* M;$
 $x \#* \Psi; x \#* xvec \implies$
 $\Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)((\nu * xvec)P')$
 $| cOpen: \llbracket \Psi \triangleright P \mapsto M(\nu * (xvec @ yvec))\langle N \rangle \prec P'; x \in supp N; x \#* xvec; x \#*$
 $yvec; x \#* M; x \#* \Psi;$
 $distinct xvec; distinct yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* yvec; yvec \#* \Psi; yvec \#* P;$
 $yvec \#* M \implies$
 $\Psi \triangleright (\nu x)P \mapsto M(\nu * (xvec @ x \# yvec))\langle N \rangle \prec P'$
 $| cBrOpen: \llbracket \Psi \triangleright P \mapsto \downarrow M(\nu * (xvec @ yvec))\langle N \rangle \prec P'; x \in supp N; x \#* xvec;$
 $x \#* yvec; x \#* M; x \#* \Psi;$
 $distinct xvec; distinct yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* yvec; yvec \#* \Psi; yvec \#* P;$
 $yvec \#* M \implies$
 $\Psi \triangleright (\nu x)P \mapsto \downarrow M(\nu * (xvec @ x \# yvec))\langle N \rangle \prec P'$
 $| cScope: \llbracket \Psi \triangleright P \mapsto \alpha \prec P'; x \#* \Psi; x \#* \alpha; bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#*$
 $(subject \alpha); distinct(bn \alpha) \implies \Psi \triangleright (\nu x)P \mapsto \alpha \prec ((\nu x)P')$
 $| Bang: \llbracket \Psi \triangleright P \parallel !P \mapsto Rs; guarded P \implies \Psi \triangleright !P \mapsto Rs$

abbreviation

$semanticsBottomJudge (- \mapsto - [50, 50] 50) \mathbf{where} P \mapsto Rs \equiv \mathbf{1} \triangleright P \mapsto Rs$

equivariance $env.semantics$

nominal-inductive2 *env.semantics*

```
avoids cInput: set xvec
| cBrInput: set xvec
| cPar1: set  $A_Q \cup \text{set}(bn \ \alpha)$ 
| cPar2: set  $A_P \cup \text{set}(bn \ \alpha)$ 
| cComm1: set  $A_P \cup \text{set } A_Q \cup \text{set } xvec$ 
| cComm2: set  $A_P \cup \text{set } A_Q \cup \text{set } xvec$ 
| cBrMerge: set  $A_P \cup \text{set } A_Q$ 
| cBrComm1: set  $A_P \cup \text{set } A_Q \cup \text{set } xvec$ 
| cBrComm2: set  $A_P \cup \text{set } A_Q \cup \text{set } xvec$ 
| cBrClose:  $\{x\} \cup \text{set } xvec$ 
| cOpen:  $\{x\} \cup \text{set } xvec \cup \text{set } yvec$ 
| cBrOpen:  $\{x\} \cup \text{set } xvec \cup \text{set } yvec$ 
| cScope:  $\{x\} \cup \text{set}(bn \ \alpha)$ 
apply –
apply(force intro: substTerm.subst4Chain subst4Chain simp add:
abs-fresh residualFresh)+
apply(force intro: substTerm.subst4Chain subst4Chain simp
add: abs-fresh residualFresh boundOutputFresh boundOutputFreshSet fresh-star-def
resChainFresh)+
done
```

lemma *nilTrans1*:

```
fixes  $\Psi$  :: 'b
and  $M$  :: 'a
and xvec :: name list
and  $N$  :: 'a
and  $P$  :: ('a, 'b, 'c) psi
```

assumes $\Psi \triangleright \mathbf{0} \mapsto M(\nu*xvec)\langle N \rangle \prec P$

shows *False*

```
using assms
apply –
by (ind-cases  $\Psi \triangleright \mathbf{0} \mapsto M(\nu*xvec)\langle N \rangle \prec P$ )
```

lemma *nilTrans1'*:

```
fixes  $\Psi$  :: 'b
and  $M$  :: 'a
and xvec :: name list
and  $N$  :: 'a
and  $P$  :: ('a, 'b, 'c) psi
```

assumes $\Psi \triangleright \mathbf{0} \mapsto iM(\nu*xvec)\langle N \rangle \prec P$

shows *False*

```
using assms
apply –
```

by (*ind-cases* $\Psi \triangleright \mathbf{0} \mapsto \mathfrak{J}M(\nu*xvec)\langle N \rangle \prec P$)

lemma *nilTrans2*:

fixes $\Psi :: 'b$
and $R_s :: ('a, 'b, 'c)$ *residual*

assumes $\Psi \triangleright \mathbf{0} \mapsto R_s$

shows *False*

using *assms*
apply(*cases rule: semantics.cases*)
by(*auto simp add: residualInject*)**+**

lemma *nilTrans3*:

fixes $\Psi :: 'b$
and $M :: 'a$
and $M' :: 'a$
and $xvec :: \text{name list}$
and $yvec :: \text{name list}$
and $N :: 'a$
and $N' :: 'a$
and $P :: ('a, 'b, 'c)$ *psi*
and $P' :: ('a, 'b, 'c)$ *psi*

assumes $\Psi \triangleright M(\lambda*xvec N).P \mapsto M'(\nu*yvec)\langle N' \rangle \prec P'$

shows *False*

using *assms*
apply –
by(*ind-cases* $\Psi \triangleright M(\lambda*xvec N).P \mapsto M'(\nu*yvec)\langle N' \rangle \prec P'$) (*auto simp add: residualInject*)

lemma *nilTrans3'*:

fixes $\Psi :: 'b$
and $M :: 'a$
and $M' :: 'a$
and $xvec :: \text{name list}$
and $yvec :: \text{name list}$
and $N :: 'a$
and $N' :: 'a$
and $P :: ('a, 'b, 'c)$ *psi*
and $P' :: ('a, 'b, 'c)$ *psi*

assumes $\Psi \triangleright M(\lambda*xvec N).P \mapsto \mathfrak{J}M'(\nu*yvec)\langle N' \rangle \prec P'$

shows *False*

using *assms*
apply –
by(*ind-cases* $\Psi \triangleright M(\lambda*xvec N).P \mapsto \mathfrak{J}M'(\nu*yvec)\langle N' \rangle \prec P'$) (*auto simp add:*

residualInject)

lemma *nilTrans4*:

fixes Ψ :: 'b
and R_s :: ('a, 'b, 'c) *residual*

assumes $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto \tau \prec P'$

shows *False*

using *assms*
apply(*cases rule: semantics.cases*)
by(*auto simp add: residualInject*)+

lemma *nilTrans5*:

fixes Ψ :: 'b
fixes Ψ' :: 'b
and M :: 'a
and $xvec$:: *name list*
and N :: 'a
and P :: ('a, 'b, 'c) *psi*

assumes $\Psi \triangleright \{\Psi'\} \mapsto M(\nu * xvec)\langle N \rangle \prec P$

shows *False*

using *assms*
apply –
by(*ind-cases* $\Psi \triangleright \{\Psi'\} \mapsto M(\nu * xvec)\langle N \rangle \prec P$)

lemma *nilTrans5'*:

fixes Ψ :: 'b
fixes Ψ' :: 'b
and M :: 'a
and $xvec$:: *name list*
and N :: 'a
and P :: ('a, 'b, 'c) *psi*

assumes $\Psi \triangleright \{\Psi'\} \mapsto \mathit{i}M(\nu * xvec)\langle N \rangle \prec P$

shows *False*

using *assms*
apply –
by(*ind-cases* $\Psi \triangleright \{\Psi'\} \mapsto \mathit{i}M(\nu * xvec)\langle N \rangle \prec P$)

lemma *nilTrans6*:

fixes Ψ :: 'b
and R_s :: ('a, 'b, 'c) *residual*

assumes $\Psi \triangleright \{\Psi'\} \mapsto R_s$

shows *False*
using *assms*
apply(*cases rule: semantics.cases*)
by(*auto simp add: residualInject*)+

lemma *nilTrans[dest]*:
fixes $\Psi :: 'b$
and $R_s :: ('a, 'b, 'c)$ *residual*
and $M :: 'a$
and $xvec :: \text{name list}$
and $N :: 'a$
and $P :: ('a, 'b, 'c)$ *psi*
and $K :: 'a$
and $yvec :: \text{name list}$
and $N' :: 'a$
and $P' :: ('a, 'b, 'c)$ *psi*
and $CsP :: ('c \times ('a, 'b, 'c)$ *psi*) *list*
and $\Psi' :: 'b$

shows $\Psi \triangleright \mathbf{0} \mapsto R_s \Longrightarrow \text{False}$
and $\Psi \triangleright M(\lambda xvec N).P \mapsto K(\nu*yvec)\langle N' \rangle \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright M(\lambda xvec N).P \mapsto_i K(\nu*yvec)\langle N' \rangle \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright M(\lambda xvec N).P \mapsto_\tau \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright M\langle N \rangle.P \mapsto K\langle N' \rangle \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright M\langle N \rangle.P \mapsto_i K\langle N' \rangle \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright M\langle N \rangle.P \mapsto_\tau \prec P' \Longrightarrow \text{False}$
and $\Psi \triangleright \{\Psi'\} \mapsto R_s \Longrightarrow \text{False}$
apply –
apply(*rule nilTrans2*)
apply *assumption*
apply(*cases rule: semantics.cases*) **apply**(*force simp add: residualInject*) +
apply(*cases rule: semantics.cases*) **apply**(*force simp add: residualInject*) +
apply(*rule nilTrans4*)
apply *assumption*
apply(*cases rule: semantics.cases*) **apply**(*force simp add: residualInject*) +
apply(*cases rule: semantics.cases*) **apply**(*force simp add: residualInject*) +
apply(*cases rule: semantics.cases*) **apply**(*force simp add: residualInject*) +
apply(*rule nilTrans6*)
by *assumption*

lemma *residualEq*:
fixes $\alpha :: 'a$ *action*
and $P :: ('a, 'b, 'c)$ *psi*
and $\beta :: 'a$ *action*
and $Q :: ('a, 'b, 'c)$ *psi*

assumes $\alpha \prec P = \beta \prec Q$
and $bn \alpha \#* (bn \beta)$
and $distinct(bn \alpha)$

```

and distinct(bn β)
and bn α #* (α < P)
and bn β #* (β < Q)

obtains p where set p ⊆ set(bn α) × set(bn(p · α)) and distinctPerm p and β
= p · α and Q = p · P and bn α #* β and bn α #* Q and bn(p · α) #* α and
bn(p · α) #* P
  using assms
proof(nominal-induct α rule: action.strong-induct)
  case(In M N)
  then show ?case by(simp add: residualInject)
next
  case(BrIn M N)
  then show ?case by(simp add: residualInject)
next
  case(Out M xvec N)
  then show ?case
    using boundOutputChainEq'' by(force simp add: residualInject)
next
  case(BrOut M xvec N)
  then show ?case
    using boundOutputChainEq'' by(force simp add: residualInject)
next
  case Tau
  then show ?case by(simp add: residualInject)
qed

lemma semanticsInduct[consumes 3, case-names cAlpha cInput cBrInput cOutput
cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2
cBrClose cOpen cBrOpen cScope cBang]:
  fixes Ψ      :: 'b
  and P        :: ('a, 'b, 'c) psi
  and α        :: 'a action
  and P'       :: ('a, 'b, 'c) psi
  and Prop     :: 'f::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒
                'a action ⇒ ('a, 'b, 'c) psi ⇒ bool
  and C        :: 'f::fs-name

assumes Ψ ▷ P ⟶ α < P'
and bn α #* (subject α)
and distinct(bn α)
and rAlpha: ∧Ψ P α P' p C. [bn α #* Ψ; bn α #* P; bn α #* (subject α);
                               bn α #* C; bn α #* (bn(p · α));
                               set p ⊆ set(bn α) × set(bn(p · α)); distinctPerm p;
                               (bn(p · α)) #* α; (bn(p · α)) #* P'; Prop C Ψ P α
P] ⇒
                               Prop C Ψ P (p · α) (p · P')
and rInput: ∧Ψ M K xvec N Tvec P C.
                [Ψ ⊢ M ↔ K; distinct xvec; set xvec ⊆ supp N;

```

$$\begin{aligned}
& \text{length } xvec = \text{length } Tvec; xvec \#* \Psi; \\
& xvec \#* M; xvec \#* K; xvec \#* C \implies \\
& \text{Prop } C \Psi (M(\lambda*xvec N).P) \\
& \quad (K(\!(N[xvec::=Tvec]\!)) (P[xvec::=Tvec])) \\
\text{and } rBrInput: & \wedge \Psi K M xvec N Tvec P C. \\
& \llbracket \Psi \vdash K \succeq M; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N; \\
& \text{length } xvec = \text{length } Tvec; xvec \#* \Psi; \\
& xvec \#* M; xvec \#* K; xvec \#* C \implies \\
& \text{Prop } C \Psi (M(\lambda*xvec N).P) \\
& \quad (\iota K(\!(N[xvec::=Tvec]\!)) (P[xvec::=Tvec])) \\
\text{and } rOutput: & \wedge \Psi M K N P C. \llbracket \Psi \vdash M \leftrightarrow K \rrbracket \implies \text{Prop } C \Psi (M\langle N \rangle.P) \\
(K\langle N \rangle) & P \\
\text{and } rBrOutput: & \wedge \Psi M K N P C. \llbracket \Psi \vdash M \preceq K \rrbracket \implies \text{Prop } C \Psi (M\langle N \rangle.P) \\
(\iota K\langle N \rangle) & P \\
\text{and } rCase: & \wedge \Psi P \alpha P' \varphi Cs C. \llbracket \Psi \triangleright P \mapsto \alpha \prec P'; \wedge C. \text{Prop } C \Psi P \alpha P'; \\
(\varphi, P) \in \text{set } Cs; & \Psi \vdash \varphi; \text{guarded } P \rrbracket \implies \\
& \text{Prop } C \Psi (\text{Cases } Cs) \alpha P' \\
\text{and } rPar1: & \wedge \Psi \Psi_Q P \alpha P' A_Q Q C. \\
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct} \\
A_Q; & \\
& \wedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P \alpha P'; \\
& A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* P'; A_Q \#* C; \\
\text{distinct}(bn \alpha); & bn \alpha \#* Q; \\
& bn \alpha \#* \Psi; bn \alpha \#* \Psi_Q; bn \alpha \#* P; bn \alpha \#* \text{subject } \alpha; bn \alpha \#* C \rrbracket \\
\implies & \\
& \text{Prop } C \Psi (P \parallel Q) \alpha (P' \parallel Q) \\
\text{and } rPar2: & \wedge \Psi \Psi_P Q \alpha Q' A_P P C. \\
& \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct} \\
A_P; & \\
& \wedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q \alpha Q'; \\
& A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* Q'; A_P \#* C; \\
\text{distinct}(bn \alpha); & bn \alpha \#* Q; \\
& bn \alpha \#* \Psi; bn \alpha \#* \Psi_P; bn \alpha \#* P; bn \alpha \#* \text{subject } \alpha; bn \alpha \#* C \rrbracket \\
\implies & \\
& \text{Prop } C \Psi (P \parallel Q) \alpha (P \parallel Q') \\
\text{and } rComm1: & \wedge \Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q C. \\
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M\langle N \rangle \prec P'; \wedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P (M\langle N \rangle) \\
P'; & \\
& \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu*xvec)\langle N \rangle \prec Q'; \wedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q \\
(K(\nu*xvec)\langle N \rangle) & Q'; \\
& \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q; \\
& \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; \\
& A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; \\
& A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P; \\
& A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q'; \\
\text{distinct } xvec; & \\
& A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* \\
M; &
\end{aligned}$$

$xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) (\tau) (\nu*xvec)(P' \parallel Q')$
and $rComm2: \bigwedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P$
 $(M(\nu*xvec)\langle N \rangle) P';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K\langle N \rangle \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (K\langle N \rangle)$
 $Q';$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $distinct xvec;$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#*$
 $M;$
 $xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) (\tau) (\nu*xvec)(P' \parallel Q')$
and $rBrMerge: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM\langle N \rangle \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P$
 $(iM\langle N \rangle) P';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM\langle N \rangle \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q$
 $(iM\langle N \rangle) Q';$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C \implies$
 $Prop C \Psi (P \parallel Q) (iM\langle N \rangle) (P' \parallel Q')$
and $rBrComm1: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM\langle N \rangle \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P (iM\langle N \rangle)$
 $P';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(\nu*xvec)\langle N \rangle \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P)$
 $Q (iM(\nu*xvec)\langle N \rangle) Q';$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; distinct xvec;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) (iM(\nu*xvec)\langle N \rangle) (P' \parallel Q')$
and $rBrComm2: \bigwedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q)$
 $P (iM(\nu*xvec)\langle N \rangle) P';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM\langle N \rangle \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (iM\langle N \rangle)$

Q' ;

extractFrame $Q = \langle A_Q, \Psi_Q \rangle$; *distinct* A_Q ;
 $A_P \#* \Psi$; $A_P \#* \Psi_Q$; $A_P \#* P$; $A_P \#* N$; $A_P \#* P'$;
 $A_P \#* Q$; $A_P \#* Q'$; $A_P \#* A_Q$; $A_P \#* xvec$; $A_Q \#* \Psi$; $A_Q \#* \Psi_P$;
 $A_Q \#* P$; $A_Q \#* N$; $A_Q \#* P'$; $A_Q \#* Q$; $A_Q \#* Q'$; *distinct* $xvec$;
 $A_P \#* M$; $A_Q \#* M$; $xvec \#* M$;
 $A_Q \#* xvec$; $xvec \#* \Psi$; $xvec \#* \Psi_P$; $xvec \#* \Psi_Q$; $xvec \#* P$;
 $xvec \#* Q$; $A_P \#* C$; $A_Q \#* C$; $xvec \#* C \rrbracket \Longrightarrow$
 $Prop\ C\ \Psi\ (P \parallel Q)\ (iM(\nu*xvec)\langle N \rangle)\ (P' \parallel Q')$

and *rBrClose*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C.$
 $\llbracket \Psi \triangleright P \longmapsto iM(\nu*xvec)\langle N \rangle \prec P';$
 $\bigwedge C.\ Prop\ C\ \Psi\ P\ (iM(\nu*xvec)\langle N \rangle)\ P';$
 $x \in \text{supp}\ M;$
distinct $xvec$; $xvec \#* \Psi$; $xvec \#* P$;
 $xvec \#* M;$
 $x \# \Psi$; $x \# xvec \rrbracket \Longrightarrow$
 $Prop\ C\ \Psi\ ((\nu x)P)\ (\tau)\ ((\nu x)((\nu*xvec)P'))$

and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
 $\llbracket \Psi \triangleright P \longmapsto M(\nu*(xvec@yvec))\langle N \rangle \prec P';\ x \in \text{supp}\ N;$
 $\bigwedge C.\ Prop\ C\ \Psi\ P\ (M(\nu*(xvec@yvec))\langle N \rangle)\ P';$
 $x \# \Psi$; $x \# M$; $x \# xvec$; $x \# yvec$; $xvec \#* \Psi$; $xvec \#* P$; $xvec \#* M$;
distinct $xvec$; *distinct* $yvec$;
 $yvec \#* \Psi$; $yvec \#* P$; $yvec \#* M$; $yvec \#* C$; $x \# C$; $xvec \#* C \rrbracket \Longrightarrow$
 $Prop\ C\ \Psi\ ((\nu x)P)\ (M(\nu*(xvec@x\#yvec))\langle N \rangle)\ P'$

and *rBrOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
 $\llbracket \Psi \triangleright P \longmapsto iM(\nu*(xvec@yvec))\langle N \rangle \prec P';\ x \in \text{supp}\ N;$
 $\bigwedge C.\ Prop\ C\ \Psi\ P\ (iM(\nu*(xvec@yvec))\langle N \rangle)\ P';$
 $x \# \Psi$; $x \# M$; $x \# xvec$; $x \# yvec$; $xvec \#* \Psi$; $xvec \#* P$; $xvec \#* M$;
distinct $xvec$; *distinct* $yvec$;
 $yvec \#* \Psi$; $yvec \#* P$; $yvec \#* M$; $yvec \#* C$; $x \# C$; $xvec \#* C \rrbracket \Longrightarrow$
 $Prop\ C\ \Psi\ ((\nu x)P)\ (iM(\nu*(xvec@x\#yvec))\langle N \rangle)\ P'$

and *rScope*: $\bigwedge \Psi\ P\ \alpha\ P'\ x\ C.$
 $\llbracket \Psi \triangleright P \longmapsto \alpha \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P';$
 $x \# \Psi$; $x \# \alpha$; $bn\ \alpha \#* \Psi$;
 $bn\ \alpha \#* P$; $bn\ \alpha \#* (\text{subject}\ \alpha)$; $x \# C$; $bn\ \alpha \#* C$; *distinct*($bn\ \alpha$) \rrbracket

\Longrightarrow

$Prop\ C\ \Psi\ ((\nu x)P)\ \alpha\ ((\nu x)P')$

and *rBang*: $\bigwedge \Psi\ P\ \alpha\ P'\ C.$
 $\llbracket \Psi \triangleright P \parallel !P \longmapsto \alpha \prec P';\ \text{guarded}\ P;$
 $\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ \alpha$
 $P \rrbracket \Longrightarrow$
 $Prop\ C\ \Psi\ (!P)\ \alpha\ P'$

shows $Prop\ C\ \Psi\ P\ \alpha\ P'$

using $\langle \Psi \triangleright P \longmapsto \alpha \prec P' \rangle$ $\langle bn\ \alpha \#* (\text{subject}\ \alpha) \rangle$ $\langle \text{distinct}(bn\ \alpha) \rangle$

proof(*nominal-induct* $x\exists == \alpha \prec P'$ *avoiding*: $\alpha\ C$ *arbitrary*: P' *rule*: *semantics.strong-induct*)

case(*cInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ \alpha\ C\ P'$)

then show *?case* **by**(*force* *intro*: *rInput* *simp* *add*: *residualInject*)

next

```

case(cBrInput  $\Psi$   $M$   $K$  xvec  $N$  Tvec  $P$   $\alpha$   $C$   $P'$ )
then show ?case
  by(force simp add: rBrInput residualInject)
next
case(Output  $\Psi$   $M$   $K$   $N$   $P$   $\alpha$   $C$   $P'$ )
then show ?case by(force intro: rOutput simp add: residualInject)
next
case(BrOutput  $\Psi$   $M$   $K$   $N$   $P$   $\alpha$   $C$   $P'$ )
then show ?case by(force intro: rBrOutput simp add: residualInject)
next
case(Case  $\Psi$   $P$   $\varphi$   $Cs$   $\alpha$   $C$   $P'$ )
then show ?case by(auto intro: rCase)
next
case(cPar1  $\Psi$   $\Psi_Q$   $P$   $\alpha$   $P'$   $Q$   $A_Q$   $\alpha'$   $C$   $P''$ )
note  $\langle \alpha \prec (P' \parallel Q) = \alpha' \prec P'' \rangle$ 
moreover from  $\langle bn \ \alpha \ \#* \ \alpha' \rangle$  have  $bn \ \alpha \ \#* \ (bn \ \alpha')$  by auto
moreover note  $\langle distinct \ (bn \ \alpha) \rangle \langle distinct \ (bn \ \alpha') \rangle$ 
moreover from  $\langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha' \ \#* \ subject \ \alpha' \rangle$ 
have  $bn \ \alpha \ \#* \ (\alpha \prec P' \parallel Q)$  and  $bn \ \alpha' \ \#* \ (\alpha' \prec P'')$  by simp+
ultimately obtain  $p$  where  $S: (set \ p) \subseteq (set \ (bn \ \alpha)) \times (set \ (bn \ (p \cdot \alpha)))$  and
distinctPerm p
  and  $\alpha Eq: \alpha' = p \cdot \alpha$  and  $P' eq: P'' = p \cdot (P' \parallel Q)$  and  $(bn \ (p \cdot \alpha)) \ \#* \ \alpha$ 
  and  $(bn \ (p \cdot \alpha)) \ \#* \ (P' \parallel Q)$ 
  by(rule residualEq)

note  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle \langle extractFrame \ Q = \langle A_Q, \Psi_Q \rangle \rangle \langle distinct \ A_Q \rangle$ 
moreover from  $\langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle distinct \ (bn \ \alpha) \rangle$ 
have  $\bigwedge C. Prop \ C \ (\Psi \otimes \Psi_Q) \ P \ \alpha \ P'$  by(metis cPar1)
moreover note  $\langle A_Q \ \#* \ P \rangle \langle A_Q \ \#* \ Q \rangle \langle A_Q \ \#* \ \Psi \rangle \langle A_Q \ \#* \ \alpha \rangle \langle A_Q \ \#* \ P' \rangle \langle A_Q \ \#* \ C \rangle$ 
   $\langle bn \ \alpha \ \#* \ Q \rangle \langle distinct \ (bn \ \alpha) \rangle \langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ \Psi_Q \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle$ 
ultimately have  $Prop \ C \ \Psi \ (P \parallel Q) \ \alpha \ (P' \parallel Q)$ 
  by(metis rPar1)

with  $\langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle \langle bn \ \alpha \ \#* \ bn \ \alpha' \rangle$ 
   $S \langle distinctPerm \ p \rangle \langle bn \ (p \cdot \alpha) \ \#* \ \alpha \rangle \langle bn \ (p \cdot \alpha) \ \#* \ (P' \parallel Q) \rangle \langle A_Q \ \#* \ C \rangle$ 
have  $Prop \ C \ \Psi \ (P \parallel Q) \ (p \cdot \alpha) \ (p \cdot (P' \parallel Q))$ 
  by  $-$  (rule rAlpha, auto)
with  $\alpha Eq \ P' eq \ \langle distinctPerm \ p \rangle$  show ?case by simp
next
case(cPar2  $\Psi$   $\Psi_P$   $Q$   $\alpha$   $Q'$   $P$   $A_P$   $\alpha'$   $C$   $Q''$ )
note  $\langle \alpha \prec (P \parallel Q') = \alpha' \prec Q'' \rangle$ 
moreover from  $\langle bn \ \alpha \ \#* \ \alpha' \rangle$  have  $bn \ \alpha \ \#* \ (bn \ \alpha')$  by auto
moreover note  $\langle distinct \ (bn \ \alpha) \rangle \langle distinct \ (bn \ \alpha') \rangle$ 
moreover from  $\langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha' \ \#* \ subject \ \alpha' \rangle$ 
have  $bn \ \alpha \ \#* \ (\alpha \prec P \parallel Q')$  and  $bn \ \alpha' \ \#* \ (\alpha' \prec Q'')$  by simp+
ultimately obtain  $p$  where  $S: (set \ p) \subseteq (set \ (bn \ \alpha)) \times (set \ (bn \ (p \cdot \alpha)))$  and
distinctPerm p

```

and $\alpha Eq: \alpha' = p \cdot \alpha$ **and** $Q' eq: Q'' = p \cdot (P \parallel Q')$ **and** $(bn(p \cdot \alpha)) \#* \alpha$
and $(bn(p \cdot \alpha)) \#* (P \parallel Q')$
by(*rule residualEq*)

note $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle$ $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$ $\langle distinct A_P \rangle$
moreover from $\langle bn \alpha \#* subject \alpha \rangle$ $\langle distinct(bn \alpha) \rangle$
have $\wedge C. Prop C (\Psi \otimes \Psi_P) Q \alpha Q'$ **by**(*auto intro: cPar2*)

moreover note $\langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* Q' \rangle \langle A_P \#* C \rangle$
 $\langle bn \alpha \#* Q \rangle \langle distinct(bn \alpha) \rangle \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* \Psi_P \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* C \rangle$
ultimately have $Prop C \Psi (P \parallel Q) \alpha (P \parallel Q')$
by(*metis rPar2*)
with $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* C \rangle \langle bn \alpha \#* (bn \alpha') \rangle S \langle distinctPerm p \rangle \langle bn(p \cdot \alpha) \#* \alpha \rangle \langle bn(p \cdot \alpha) \#* (P \parallel Q') \rangle$
have $Prop C \Psi (P \parallel Q) (p \cdot \alpha) (p \cdot (P \parallel Q'))$
by – (*rule rAlpha, auto*)
with $\alpha Eq Q' eq \langle distinctPerm p \rangle$ **show** $?case$ **by** *simp*
next
case(*cComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q \alpha C P''$)
then have $Prop C \Psi (P \parallel Q) (\tau) ((\nu * xvec)(P' \parallel Q'))$
by(*fastforce intro: rComm1*)
then show $?case$ **using** $\langle \tau \prec ((\nu * xvec)(P' \parallel Q')) = \alpha \prec P'' \rangle$
by(*simp add: residualInject*)
next
case(*cComm2* $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q \alpha C P''$)
then have $Prop C \Psi (P \parallel Q) (\tau) ((\nu * xvec)(P' \parallel Q'))$
by(*fastforce intro: rComm2*)
then show $?case$ **using** $\langle \tau \prec ((\nu * xvec)(P' \parallel Q')) = \alpha \prec P'' \rangle$
by(*simp add: residualInject*)
next
case (*cBrMerge* $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q \alpha C P''$)
then show $?case$ **by**(*simp add: rBrMerge residualInject*)
next
case(*cBrComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q \alpha C P''$)
from *cBrComm1* $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle xvec \#* M \rangle$ **have** $Prop C \Psi (P \parallel Q)$
 $(\text{j}M(\nu * xvec)\langle N \rangle) (P' \parallel Q')$
by(*fastforce intro: rBrComm1*)

note $\langle \text{j}M(\nu * xvec)\langle N \rangle \prec P' \parallel Q' = \alpha \prec P'' \rangle$
moreover from $\langle xvec \#* \alpha \rangle$ **have** $bn (\text{j}M(\nu * xvec)\langle N \rangle) \#* (bn \alpha)$ **by** *simp*
moreover from $\langle distinct xvec \rangle$ **have** $distinct (bn (\text{j}M(\nu * xvec)\langle N \rangle))$ **by** *simp*
moreover note $\langle distinct (bn \alpha) \rangle$
moreover from $\langle xvec \#* M \rangle$ **have** $bn (\text{j}M(\nu * xvec)\langle N \rangle) \#* (\text{j}M(\nu * xvec)\langle N \rangle \prec P' \parallel Q')$ **by** *simp*
moreover from $\langle bn \alpha \#* subject \alpha \rangle$ **have** $bn \alpha \#* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** $S: (set p) \subseteq (set(bn (\text{j}M(\nu * xvec)\langle N \rangle))) \times (set(bn(p \cdot (\text{j}M(\nu * xvec)\langle N \rangle))))$ **and** $distinctPerm p$

and $\alpha Eq: \alpha = p \cdot (iM(\nu*xvec)\langle N \rangle)$ **and** $P'eq: P'' = p \cdot (P' \parallel Q')$ **and** $(bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (iM(\nu*xvec)\langle N \rangle)$
and $(bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q')$
by(*rule residualEq*)

from $\langle xvec \#* \Psi \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* \Psi$ **by** *simp*
moreover from $\langle xvec \#* P \rangle \langle xvec \#* Q \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (P \parallel Q)$
by *simp*
moreover from $\langle xvec \#* M \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* subject (iM(\nu*xvec)\langle N \rangle)$
by *simp*
moreover from $\langle xvec \#* C \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* C$ **by** *simp*
moreover from $\langle (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (iM(\nu*xvec)\langle N \rangle) \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* bn (p \cdot (iM(\nu*xvec)\langle N \rangle))$ **by** *simp*
moreover note $\langle (set p) \subseteq (set (bn (iM(\nu*xvec)\langle N \rangle))) \times (set (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \rangle$
moreover note $\langle distinctPerm p \rangle$
moreover note $\langle (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (iM(\nu*xvec)\langle N \rangle) \rangle$
moreover note $\langle (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q') \rangle$
moreover note $\langle Prop C \Psi (P \parallel Q) (iM(\nu*xvec)\langle N \rangle) (P' \parallel Q') \rangle$

ultimately have *propEqvt: Prop C Ψ (P ∥ Q) (p · (iM(ν*xvec)⟨N⟩)) (p · (P' ∥ Q'))* **by**(*rule rAlpha*)

then show *?case by (simp add: αEq P'eq propEqvt)*
next

case(*cBrComm2 Ψ Ψ_Q P M xvec N P' A_P Ψ_P Q Q' A_Q α C P''*)
from *cBrComm2* $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle xvec \#* M \rangle$ **have** *Prop C Ψ (P ∥ Q)*
 $(iM(\nu*xvec)\langle N \rangle) (P' \parallel Q')$
by(*fastforce intro: rBrComm2*)

note $\langle iM(\nu*xvec)\langle N \rangle \prec P' \parallel Q' = \alpha \prec P'' \rangle$
moreover from $\langle xvec \#* \alpha \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (bn \alpha)$ **by** *simp*
moreover from $\langle distinct xvec \rangle$ **have** $distinct (bn (iM(\nu*xvec)\langle N \rangle))$ **by** *simp*
moreover note $\langle distinct (bn \alpha) \rangle$
moreover from $\langle xvec \#* M \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (iM(\nu*xvec)\langle N \rangle \prec P' \parallel Q')$ **by** *simp*
moreover from $\langle bn \alpha \#* subject \alpha \rangle$ **have** $bn \alpha \#* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** $S: (set p) \subseteq (set (bn (iM(\nu*xvec)\langle N \rangle))) \times (set (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))))$ **and** *distinctPerm p*
and $\alpha Eq: \alpha = p \cdot (iM(\nu*xvec)\langle N \rangle)$ **and** $P'eq: P'' = p \cdot (P' \parallel Q')$ **and** $(bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (iM(\nu*xvec)\langle N \rangle)$
and $(bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q')$
by(*rule residualEq*)

from $\langle xvec \#* \Psi \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* \Psi$ **by** *simp*
moreover from $\langle xvec \#* P \rangle \langle xvec \#* Q \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (P \parallel Q)$
by *simp*
moreover from $\langle xvec \#* M \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* subject (iM(\nu*xvec)\langle N \rangle)$
by *simp*
moreover from $\langle xvec \#* C \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* C$ **by** *simp*

```

moreover from ⟨(bn(p · (iM(ν*xvec)⟨N⟩))) #* (iM(ν*xvec)⟨N⟩)⟩ have bn
(iM(ν*xvec)⟨N⟩) #* bn (p · (iM(ν*xvec)⟨N⟩)) by simp
moreover note ⟨(set p) ⊆ (set(bn (iM(ν*xvec)⟨N⟩))) × (set(bn(p · (iM(ν*xvec)⟨N⟩))))⟩
moreover note ⟨distinctPerm p⟩
moreover note ⟨(bn(p · (iM(ν*xvec)⟨N⟩))) #* (iM(ν*xvec)⟨N⟩)⟩
moreover note ⟨(bn(p · (iM(ν*xvec)⟨N⟩))) #* (P' || Q')⟩
moreover note ⟨Prop C Ψ (P || Q) (iM(ν*xvec)⟨N⟩) (P' || Q')⟩

ultimately have propEqvt: Prop C Ψ (P || Q) (p · (iM(ν*xvec)⟨N⟩)) (p · (P'
|| Q')) by(rule rAlpha)

then show ?case by (simp add: αEq P'eq propEqvt)
next
case (cBrClose Ψ P M xvec N P' x α C P'')
then have Prop C Ψ ((νx)P) (τ) ((νx)((ν*xvec)P'))
by(fastforce intro: rBrClose)
then show ?case using ⟨τ < (νx)((ν*xvec)P') = α < P''⟩
by(simp add: residualInject)
next
case(cOpen Ψ P M xvec yvec N P' x α C P'')
note ⟨M(ν*(xvec@x#yvec)⟨N⟩) < P' = α < P''⟩
moreover from ⟨xvec #* α⟩ ⟨x # α⟩ ⟨yvec #* α⟩ have (xvec@x#yvec) #* (bn α)
by auto
moreover from ⟨xvec #* yvec⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩
have distinct(xvec@x#yvec)
by(auto simp add: fresh-star-def) (simp add: fresh-def name-list-supp)
moreover note ⟨distinct(bn α)⟩
moreover from ⟨xvec #* M⟩ ⟨x # M⟩ ⟨yvec #* M⟩ have (xvec@x#yvec) #* M
by auto
then have (xvec@x#yvec) #* (M(ν*(xvec@x#yvec)⟨N⟩) < P') by auto
moreover from ⟨bn α #* subject α⟩ have bn α #* (α < P'') by simp
ultimately obtain p where S: (set p) ⊆ (set(xvec@x#yvec)) × (set(p · (xvec@x#yvec)))
and distinctPerm p
and αeq: α = (p · M)(ν*(p · (xvec@x#yvec)))⟨(p · N)⟩ and P'eq: P'' = (p ·
P')
and A: (xvec@x#yvec) #* ((p · M)(ν*(p · (xvec@x#yvec)))⟨(p · N)⟩)
and B: (p · (xvec@x#yvec)) #* (M(ν*(xvec@x#yvec)⟨N⟩))
and C: (p · (xvec@x#yvec)) #* P'
by - (rule residualEq, (assumption | simp)+)
note ⟨Ψ ▷ P ⟶ M(ν*(xvec@yvec)⟨N⟩) < P'⟩ ⟨x ∈ (supp N)⟩

moreover {
fix C
from ⟨xvec #* M⟩ ⟨yvec #* M⟩ have (xvec@yvec) #* M by simp
moreover from ⟨distinct xvec⟩ ⟨distinct yvec⟩ ⟨xvec #* yvec⟩ have distinct(xvec@yvec)
by auto (simp add: fresh-star-def name-list-supp fresh-def)
ultimately have Prop C Ψ P (M(ν*(xvec@yvec)⟨N⟩) P') by(fastforce intro:
cOpen)
}

```

moreover note $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle xvec \# \Psi \rangle \langle xvec \# P \rangle$
 $\langle xvec \# M \rangle$
 $\langle yvec \# \Psi \rangle \langle yvec \# P \rangle \langle yvec \# M \rangle \langle yvec \# C \rangle \langle x \# C \rangle \langle xvec \# C \rangle \langle distinct$
 $xvec \rangle \langle distinct yvec \rangle$
ultimately have $Prop\ C\ \Psi\ ((\nu x)P)\ (M(\nu*(xvec@x\#yvec))\langle N \rangle)\ P'$
by(metis rOpen)

with $\langle xvec \# \Psi \rangle \langle yvec \# \Psi \rangle \langle xvec \# P \rangle \langle yvec \# P \rangle \langle xvec \# M \rangle \langle yvec \# M \rangle$
 $\langle yvec \# C \rangle\ S\ \langle distinctPerm\ p \rangle \langle x \# C \rangle \langle xvec \# C \rangle$
 $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle\ A\ B\ C$
have $Prop\ C\ \Psi\ ((\nu x)P)\ (p \cdot (M(\nu*(xvec@x\#yvec))\langle N \rangle))\ (p \cdot P')$
apply –
apply(rule rAlpha[**where** $\alpha = M(\nu*(xvec@x\#yvec))\langle N \rangle$]; clarsimp)
by (meson abs-fresh(1) abs-fresh-list-star' freshChainAppend freshSets(5) psiFreshVec(5))
with $\alpha eq\ P' eq$ **show** ?case **by** simp

next
case(cBrOpen $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ \alpha\ C\ P''$)
note $\langle iM(\nu*(xvec@x\#yvec))\langle N \rangle \prec P' = \alpha \prec P'' \rangle$
moreover from $\langle xvec \# \alpha \rangle \langle x \# \alpha \rangle \langle yvec \# \alpha \rangle$ **have** $(xvec@x\#yvec)\ \#*\ (bn\ \alpha)$
by auto
moreover from $\langle xvec \# yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle distinct\ xvec \rangle \langle distinct\ yvec \rangle$
have $distinct(xvec@x\#yvec)$
by(clarsimp simp add: fresh-star-def; safe; simp add: fresh-def name-list-supp)
moreover note $\langle distinct(bn\ \alpha) \rangle$
moreover from $\langle xvec \# M \rangle \langle x \# M \rangle \langle yvec \# M \rangle$ **have** $(xvec@x\#yvec)\ \#*\ M$
by auto
then have $(xvec@x\#yvec)\ \#*\ (iM(\nu*(xvec@x\#yvec))\langle N \rangle \prec P')$ **by** auto
moreover from $\langle bn\ \alpha\ \#*\ subject\ \alpha \rangle$ **have** $bn\ \alpha\ \#*\ (\alpha \prec P'')$ **by** simp
ultimately obtain p **where** $S: (set\ p) \subseteq (set(xvec@x\#yvec)) \times (set(p \cdot (xvec@x\#yvec)))$
and $distinctPerm\ p$
and $\alpha eq: \alpha = i(p \cdot M)(\nu*(p \cdot (xvec@x\#yvec))\langle (p \cdot N) \rangle)$ **and** $P' eq: P'' = (p \cdot$
 $P')$
and $A: (xvec@x\#yvec)\ \#*\ (i(p \cdot M)(\nu*(p \cdot (xvec@x\#yvec))\langle (p \cdot N) \rangle))$
and $B: (p \cdot (xvec@x\#yvec))\ \#*\ (iM(\nu*(xvec@x\#yvec))\langle N \rangle)$
and $C: (p \cdot (xvec@x\#yvec))\ \#*\ P'$
apply –
by(rule residualEq) (assumption|simp)+
note $\langle \Psi \triangleright P \mapsto iM(\nu*(xvec@yvec))\langle N \rangle \prec P' \rangle \langle x \in (supp\ N) \rangle$

moreover {
fix C
from $\langle xvec \# M \rangle \langle yvec \# M \rangle$ **have** $(xvec@yvec)\ \#*\ M$ **by** simp
moreover from $\langle distinct\ xvec \rangle \langle distinct\ yvec \rangle \langle xvec \# yvec \rangle$ **have** $distinct(xvec@yvec)$
by auto (simp add: fresh-star-def name-list-supp fresh-def)
ultimately have $Prop\ C\ \Psi\ P\ (iM(\nu*(xvec@yvec))\langle N \rangle)\ P'$ **by**(fastforce intro:
cBrOpen)
}

```

moreover note ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ ⟨xvec #* Ψ⟩ ⟨xvec #* P⟩
⟨xvec #* M⟩
  ⟨yvec #* Ψ⟩ ⟨yvec #* P⟩ ⟨yvec #* M⟩ ⟨yvec #* C⟩ ⟨x # C⟩ ⟨xvec #* C⟩ ⟨distinct
xvec⟩ ⟨distinct yvec⟩
ultimately have Prop C Ψ ((νx)P) (iM(ν*(xvec@x#yvec))(N)) P'
  by(metis rBrOpen)

with ⟨xvec #* Ψ⟩ ⟨yvec #* Ψ⟩ ⟨xvec #* P⟩ ⟨yvec #* P⟩ ⟨xvec #* M⟩ ⟨yvec #* M⟩
  ⟨yvec #* C⟩ S ⟨distinctPerm p⟩ ⟨x # C⟩ ⟨xvec #* C⟩
  ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ A B C
have Prop C Ψ ((νx)P) (p · (iM(ν*(xvec@x#yvec))(N))) (p · P')
  apply -
  apply(rule rAlpha[where α=iM(ν*(xvec@x#yvec))(N)]; clarsimp)
  by(meson abs-fresh(1) abs-fresh-list-star' freshChainAppend freshSets(5) psiFreshVec(5))
with αeq P'eq show ?case by simp
next
case(cScope Ψ P α P' x α' C P'')
note ⟨α < ((νx)P') = α' < P''⟩
moreover from ⟨bn α #* α'⟩ have bn α #* (bn α') by auto
moreover note ⟨distinct (bn α)⟩ ⟨distinct (bn α')⟩
moreover from ⟨bn α #* subject α⟩ ⟨bn α' #* subject α'⟩
have bn α #* (α < ((νx)P')) and bn α' #* (α' < P'') by simp+
ultimately obtain p where S: (set p) ⊆ (set (bn α)) × (set (bn (p · α))) and
distinctPerm p
  and αEq: α' = p · α and P'eq: P'' = p · ((νx)P') and (bn (p · α)) #* α
  and (bn (p · α)) #* ((νx)P')
  by(rule residualEq)

note ⟨Ψ ▷ P ⟶ α < P'⟩
moreover from ⟨bn α #* subject α⟩ ⟨distinct (bn α)⟩
have ∧C. Prop C Ψ P α P' by(fastforce intro: cScope)

moreover note ⟨x # Ψ⟩ ⟨x # α⟩ ⟨bn α #* Ψ⟩ ⟨bn α #* P⟩ ⟨bn α #* subject α⟩
  ⟨x # C⟩ ⟨bn α #* C⟩ ⟨distinct (bn α)⟩
ultimately have Prop C Ψ ((νx)P) α ((νx)P')
  by(rule rScope)
with ⟨bn α #* Ψ⟩ ⟨bn α #* P⟩ ⟨x # α⟩ ⟨bn α #* subject α⟩ ⟨bn α #* C⟩ ⟨bn α #*
(bn α')⟩ S ⟨distinctPerm p⟩ ⟨bn (p · α) #* α⟩ ⟨bn (p · α) #* ((νx)P')⟩
have Prop C Ψ ((νx)P) (p · α) (p · ((νx)P'))
  by(fastforce intro: rAlpha)
with αEq P'eq ⟨distinctPerm p⟩ show ?case by simp
next
case(Bang Ψ P α C P')
then show ?case by(fastforce intro: rBang)
qed

lemma outputInduct[consumes 1, case-names cOutput cCase cPar1 cPar2 cOpen
cScope cBang]:
  fixes Ψ :: 'b

```

and $P :: ('a, 'b, 'c) \text{psi}$
and $M :: 'a$
and $B :: ('a, 'b, 'c) \text{boundOutput}$
and $\text{Prop} :: 'f::\text{fs-name} \Rightarrow 'b \Rightarrow ('a, 'b, 'c) \text{psi} \Rightarrow$
 $'a \Rightarrow ('a, 'b, 'c) \text{boundOutput} \Rightarrow \text{bool}$
and $C :: 'f::\text{fs-name}$

assumes $\Psi \triangleright P \mapsto \text{ROut } M B$
and $r\text{Output}: \bigwedge \Psi M K N P C. [\Psi \vdash M \leftrightarrow K] \Longrightarrow \text{Prop } C \Psi (M \langle N \rangle . P) K (N \prec' P)$
and $r\text{Case}: \bigwedge \Psi P M B \varphi Cs C.$
 $[\Psi \triangleright P \mapsto (\text{ROut } M B); \bigwedge C. \text{Prop } C \Psi P M B; (\varphi, P) \in \text{set } Cs;$
 $\Psi \vdash \varphi; \text{guarded } P] \Longrightarrow$
 $\text{Prop } C \Psi (\text{Cases } Cs) M B$
and $r\text{Par1}: \bigwedge \Psi \Psi_Q P M \text{vec } N P' A_Q Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * \text{vec}) \langle N \rangle \prec P'; \text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle; \text{distinct } A_Q;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P M ((\nu * \text{vec}) \langle N \rangle \prec' P');$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M;$
 $A_Q \#* \text{vec}; A_Q \#* N; A_Q \#* P'; A_Q \#* C; \text{vec} \#* Q;$
 $\text{vec} \#* \Psi; \text{vec} \#* \Psi_Q; \text{vec} \#* P; \text{vec} \#* M; \text{vec} \#* C] \Longrightarrow$
 $\text{Prop } C \Psi (P \parallel Q) M ((\nu * \text{vec}) \langle N \rangle \prec' (P' \parallel Q))$
and $r\text{Par2}: \bigwedge \Psi \Psi_P Q M \text{vec } N Q' A_P P C.$
 $[\Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu * \text{vec}) \langle N \rangle \prec Q'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q M ((\nu * \text{vec}) \langle N \rangle \prec' Q');$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M;$
 $A_P \#* \text{vec}; A_P \#* N; A_P \#* Q'; A_P \#* C; \text{vec} \#* P;$
 $\text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* Q; \text{vec} \#* M; \text{vec} \#* C] \Longrightarrow$
 $\text{Prop } C \Psi (P \parallel Q) M ((\nu * \text{vec}) \langle N \rangle \prec' (P \parallel Q'))$
and $r\text{Open}: \bigwedge \Psi P M \text{vec } yvec N P' x C.$
 $[\Psi \triangleright P \mapsto M(\nu * (\text{vec} @ yvec)) \langle N \rangle \prec P'; x \in \text{supp } N; \bigwedge C. \text{Prop } C$
 $\Psi P M ((\nu * (\text{vec} @ yvec)) \langle N \rangle \prec' P');$
 $x \# \Psi; x \# M; x \# \text{vec}; x \# yvec; \text{vec} \#* \Psi; \text{vec} \#* P; \text{vec} \#* M;$
 $\text{vec} \#* yvec; yvec \#* \Psi; yvec \#* P; yvec \#* M; yvec \#* C; x \# C;$
 $\text{vec} \#* C] \Longrightarrow$
 $\text{Prop } C \Psi ((\nu x) P) M ((\nu * (\text{vec} @ x \# yvec)) \langle N \rangle \prec' P')$
and $r\text{Scope}: \bigwedge \Psi P M \text{vec } N P' x C.$
 $[\Psi \triangleright P \mapsto M(\nu * \text{vec}) \langle N \rangle \prec P'; \bigwedge C. \text{Prop } C \Psi P M ((\nu * \text{vec}) \langle N \rangle$
 $\prec' P');$
 $x \# \Psi; x \# M; x \# \text{vec}; x \# N; \text{vec} \#* \Psi; \text{vec} \#* P; \text{vec} \#* M;$
 $x \# C; \text{vec} \#* C] \Longrightarrow$
 $\text{Prop } C \Psi ((\nu x) P) M ((\nu * \text{vec}) \langle N \rangle \prec' (\nu x) P')$
and $r\text{Bang}: \bigwedge \Psi P M B C.$
 $[\Psi \triangleright P \parallel !P \mapsto (\text{ROut } M B); \text{guarded } P; \bigwedge C. \text{Prop } C \Psi (P \parallel$
 $!P) M B] \Longrightarrow$
 $\text{Prop } C \Psi (!P) M B$

shows $\text{Prop } C \Psi P M B$
using $\langle \Psi \triangleright P \mapsto (\text{ROut } M B) \rangle$

```

proof(nominal-induct  $\Psi P Rs== (ROut M B)$  avoiding:  $C$  arbitrary:  $B$  rule: semantics.strong-induct)
  case(cInput  $\Psi M K xvec N Tvec P C$ )
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by(simp add: residualInject)
next
  case(Output  $\Psi M K N P C$ )
  then show ?case by(force simp add: residualInject intro: rOutput)
next
  case(BrOutput  $\Psi M N P C$ )
  then show ?case by(simp add: residualInject)
next
  case(Case  $\Psi P \varphi Cs C B$ )
  then show ?case by(force intro: rCase)
next
  case(cPar1  $\Psi \Psi_Q P \alpha P' Q A_Q C$ )
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case(cPar2  $\Psi \Psi_P Q \alpha Q' P A_P C$ )
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case (cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C B$ )
  then show ?case by(simp add: residualInject)
next
  case (cBrComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C B$ )
  then show ?case by(simp add: residualInject)
next
  case(cBrClose  $\Psi P M xvec N P' C B$ )
  then show ?case by(simp add: residualInject)
next
  case(cOpen  $\Psi P M xvec yvec N P' x C B$ )
  then show ?case by(force intro: rOpen simp add: residualInject)
next
  case cBrOpen
  then show ?case by(simp add: residualInject)
next
  case(cScope  $\Psi P M \alpha P' x C$ )
  then show ?case by(force intro: rScope simp add: residualInject)

```

next

case(*Bang* Ψ *P C B*)
then show *?case* **by**(*force intro: rBang*)
qed

lemma *brOutputInduct*[*consumes 1, case-names cBrOutput cCase cPar1 cPar2 cBrComm1 cBrComm2 cBrOpen cScope cBang*]:

fixes Ψ :: 'b
and *P* :: ('a, 'b, 'c) *psi*
and *M* :: 'a
and *B* :: ('a, 'b, 'c) *boundOutput*
and *Prop* :: 'f::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) *psi* \Rightarrow
' a \Rightarrow ('a, 'b, 'c) *boundOutput* \Rightarrow *bool*
and *C* :: 'f::fs-name

assumes $\Psi \triangleright P \mapsto RBrOut\ M\ B$

and *rBrOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C. [\Psi \vdash M \preceq K] \Longrightarrow Prop\ C\ \Psi\ (M \langle N \rangle . P)\ K$
(*N* \prec' *P*)

and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ C.$

$[\Psi \triangleright P \mapsto (RBrOut\ M\ B); \bigwedge C. Prop\ C\ \Psi\ P\ M\ B; (\varphi, P) \in set$
Cs; $\Psi \vdash \varphi$; *guarded P*] \Longrightarrow

Prop C Ψ (*Cases Cs*) *M B*

and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ C.$

$[\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'; extractFrame\ Q = \langle A_Q,$
 $\Psi_Q \rangle$; *distinct A_Q*;

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ ((\nu * xvec) N \prec' P');$

A_Q # P*; *A_Q #* Q*; *A_Q #* Ψ* ; *A_Q #* M*;

A_Q # xvec*; *A_Q #* N*; *A_Q #* P'*; *A_Q #* C*; *xvec #* Q*;

xvec # Ψ* ; *xvec #* Ψ_Q* ; *xvec #* P*; *xvec #* M*; *xvec #* C*] \Longrightarrow

Prop C Ψ (*P* \parallel *Q*) *M* ($(\nu * xvec) N \prec' (P' \parallel Q)$)

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ xvec\ N\ Q'\ A_P\ P\ C.$

$[\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'; extractFrame\ P = \langle A_P,$
 $\Psi_P \rangle$; *distinct A_P*;

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\nu * xvec) N \prec' Q');$

A_P # P*; *A_P #* Q*; *A_P #* Ψ* ; *A_P #* M*;

A_P # xvec*; *A_P #* N*; *A_P #* Q'*; *A_P #* C*; *xvec #* P*;

xvec # Ψ* ; *xvec #* Ψ_P* ; *xvec #* Q*; *xvec #* M*; *xvec #* C*] \Longrightarrow

Prop C Ψ (*P* \parallel *Q*) *M* ($(\nu * xvec) N \prec' (P \parallel Q')$)

and *rBrComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$

$[\Psi \otimes \Psi_Q \triangleright P \mapsto_i M \langle N \rangle \prec P';$
 $extractFrame\ P = \langle A_P, \Psi_P \rangle$; *distinct A_P*;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'; \bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)$
 $Q\ M\ ((\nu * xvec) N \prec' Q');$

$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$; *distinct A_Q*;

A_P # Ψ* ; *A_P #* Ψ_Q* ; *A_P #* P*; *A_P #* N*; *A_P #* P'*;

A_P # Q*; *A_P #* Q'*; *A_P #* A_Q*; *A_P #* xvec*; *A_Q #* Ψ* ; *A_Q #* Ψ_P* ;

A_Q # P*; *A_Q #* N*; *A_Q #* P'*; *A_Q #* Q*; *A_Q #* Q'*; *distinct xvec*;

A_P # M*; *A_Q #* M*; *xvec #* M*;

A_Q # xvec*; *xvec #* Ψ* ; *xvec #* Ψ_P* ; *xvec #* Ψ_Q* ; *xvec #* P*;

$xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu*xvec)N \prec' (P' \parallel Q'))$
and $rBrComm2: \bigwedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M((\nu*xvec)\langle N \rangle \prec P'); \bigwedge C. Prop C (\Psi \otimes \Psi_Q)$
 $P M ((\nu*xvec)\langle N \rangle \prec' P') \rrbracket;$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\langle N \rangle \prec Q');$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; distinct xvec;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu*xvec)\langle N \rangle \prec' (P' \parallel Q'))$
and $rBrOpen: \bigwedge \Psi P M xvec yvec N P' x C.$
 $\llbracket \Psi \triangleright P \mapsto_i M((\nu*(xvec@yvec))\langle N \rangle \prec P'; x \in supp N; \bigwedge C. Prop$
 $C \Psi P M ((\nu*(xvec@yvec))\langle N \rangle \prec' P') \rrbracket;$
 $x \# \Psi; x \# M; x \# xvec; x \# yvec; xvec \#* \Psi; xvec \#* P; xvec \#* M;$
 $xvec \#* yvec; yvec \#* \Psi; yvec \#* P; yvec \#* M; yvec \#* C; x \# C;$
 $xvec \#* C \implies$
 $Prop C \Psi ((\nu x)P) M ((\nu*(xvec@x\#yvec))\langle N \rangle \prec' P')$
and $rScope: \bigwedge \Psi P M xvec N P' x C.$
 $\llbracket \Psi \triangleright P \mapsto_i M((\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. Prop C \Psi P M ((\nu*xvec)\langle N \rangle$
 $\prec' P') \rrbracket;$
 $x \# \Psi; x \# M; x \# xvec; x \# N; xvec \#* \Psi; xvec \#* P; xvec \#* M;$
 $x \# C; xvec \#* C \implies$
 $Prop C \Psi ((\nu x)P) M ((\nu*xvec)\langle N \rangle \prec' (\nu x)P')$
and $rBang: \bigwedge \Psi P M B C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto (RBrOut M B); guarded P; \bigwedge C. Prop C \Psi (P \parallel$
 $!P) M B \rrbracket \implies$
 $Prop C \Psi (!P) M B$
shows $Prop C \Psi P M B$
using $\langle \Psi \triangleright P \mapsto (RBrOut M B) \rangle$
proof(*nominal-induct* $\Psi P Rs==(RBrOut M B)$ *avoiding: C arbitrary: B rule:*
semantics.strong-induct)
case(*cInput* $\Psi M K xvec N Tvec P C$)
then show $?case$ **by**(*simp add: residualInject*)
next
case *cBrInput*
then show $?case$ **by**(*simp add: residualInject*)
next
case(*Output* $\Psi M K N P C$)
then show $?case$ **by**(*simp add: residualInject*)
next
case(*BrOutput* $\Psi M N P C$)
then show $?case$ **by**(*auto simp add: residualInject intro: rBrOutput*)
next
case(*Case* $\Psi P \varphi Cs C B$)


```

    then show ?case by(force intro: rCase)
next
  case(cPar1  $\Psi \Psi_Q P \alpha P' Q A_Q C$ )
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case(cPar2  $\Psi \Psi_P Q \alpha Q' P A_P C$ )
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case (cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C B$ )
  then show ?case by(force intro: rBrComm1 simp add: residualInject)
next
  case (cBrComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C B$ )
  then show ?case by(force intro: rBrComm2 simp add: residualInject)
next
  case(cBrClose  $\Psi P M xvec N P' C B$ )
  then show ?case by(simp add: residualInject)
next
  case(cOpen  $\Psi P M xvec yvec N P' x C B$ )
  then show ?case by(simp add: residualInject)
next
  case(cBrOpen  $\Psi P M xvec yvec N P' x C B$ )
  then show ?case by(force intro: rBrOpen simp add: residualInject)
next
  case(cScope  $\Psi P M \alpha P' x C$ )
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case(Bang  $\Psi P C B$ )
  then show ?case by(force intro: rBang)
qed

```

lemma *boundOutputBindObject*:

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $yvec$   :: name list
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $y$      :: name

```

assumes $\Psi \triangleright P \mapsto \alpha \prec P'$

```

and  $bn\ \alpha\ \#\ast\ subject\ \alpha$ 
and  $distinct(bn\ \alpha)$ 
and  $y \in set(bn\ \alpha)$ 

shows  $y \in supp(object\ \alpha)$ 
  using  $assms$ 
proof( $nominal-induct\ avoiding:\ P'\ arbitrary:\ y\ rule:\ semanticsInduct$ )
  case( $cAlpha\ \Psi\ P\ \alpha\ P'\ p\ P''\ y$ )
  from  $\langle y \in set(bn(p \cdot \alpha)) \rangle$  have  $(p \cdot y) \in (p \cdot set(bn(p \cdot \alpha)))$ 
    by( $rule\ pt-set-bij2[OF\ pt-name-inst,\ OF\ at-name-inst]$ )
  then have  $(p \cdot y) \in set(bn\ \alpha)$  using  $\langle distinctPerm\ p \rangle$ 
    by( $simp\ add:\ eqvts$ )
  then have  $(p \cdot y) \in supp(object\ \alpha)$  by( $rule\ cAlpha$ )
  then have  $(p \cdot p \cdot y) \in (p \cdot supp(object\ \alpha))$ 
    by( $rule\ pt-set-bij2[OF\ pt-name-inst,\ OF\ at-name-inst]$ )
  then show  $?case$  using  $\langle distinctPerm\ p \rangle$ 
    by( $simp\ add:\ eqvts$ )
next
  case  $cInput$ 
  then show  $?case$  by  $simp$ 
next
  case  $cBrInput$ 
  then show  $?case$  by  $simp$ 
next
  case  $cOutput$ 
  then show  $?case$  by  $simp$ 
next
  case  $cBrOutput$ 
  then show  $?case$  by  $simp$ 
next
  case  $cCase$ 
  then show  $?case$  by  $simp$ 
next
  case  $cPar1$ 
  then show  $?case$  by  $simp$ 
next
  case  $cPar2$ 
  then show  $?case$  by  $simp$ 
next
  case  $cComm1$ 
  then show  $?case$  by  $simp$ 
next
  case  $cComm2$ 
  then show  $?case$  by  $simp$ 
next
  case  $cBrMerge$ 
  then show  $?case$  by  $simp$ 
next
  case  $cBrComm1$ 

```

```

    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case cOpen
    then show ?case by (auto simp add: supp-list-cons supp-list-append supp-atm
supp-some)
  next
    case cBrOpen
    then show ?case by (auto simp add: supp-list-cons supp-list-append supp-atm
supp-some)
  next
    case cScope
    then show ?case by simp
  next
    case cBang
    then show ?case by simp
qed

```

```

lemma alphaBoundOutputChain':
  fixes yvec :: name list
    and xvec :: name list
    and B    :: ('a, 'b, 'c) boundOutput

```

```

assumes length xvec = length yvec
  and yvec #* B
  and yvec #* xvec
  and distinct yvec

```

```

shows ( $\nu^*xvec$ )B = ( $\nu^*yvec$ )([xvec yvec]  $\cdot_v$  B)
  using assms
proof (induct rule: composePermInduct)
  case cBase
  show ?case by simp
next
  case (cStep x xvec y yvec)
  then show ?case
    by (auto simp add: alphaBoundOutput[of y] eqvts)
qed

```

```

lemma alphaBoundOutputChain'':
  fixes yvec :: name list
    and xvec :: name list
    and N    :: 'a
    and P    :: ('a, 'b, 'c) psi

```

```

assumes  $\text{length } xvec = \text{length } yvec$ 
and  $yvec \#* N$ 
and  $yvec \#* P$ 
and  $yvec \#* xvec$ 
and  $\text{distinct } yvec$ 

shows  $(\nu*xvec)(N \prec' P) = (\nu*yvec)(([xvec \ yvec] \cdot_v N) \prec' ([xvec \ yvec] \cdot_v P))$ 
proof -
  from assms have  $(\nu*xvec)(N \prec' P) = (\nu*yvec)(([xvec \ yvec] \cdot_v (N \prec' P)))$ 
    by(simp add: alphaBoundOutputChain')
  then show ?thesis by simp
qed

```

```

lemma alphaDistinct:
  fixes  $xvec :: \text{name list}$ 
  and  $N :: 'a$ 
  and  $P :: ('a, 'b, 'c) \text{psi}$ 
  and  $yvec :: \text{name list}$ 
  and  $M :: 'a$ 
  and  $Q :: ('a, 'b, 'c) \text{psi}$ 

```

```

assumes  $\alpha \prec P = \beta \prec Q$ 
and  $\text{distinct}(bn \ \alpha)$ 
and  $\bigwedge x. x \in \text{set}(bn \ \alpha) \implies x \in \text{supp}(\text{object } \alpha)$ 
and  $bn \ \alpha \#* bn \ \beta$ 
and  $bn \ \alpha \#* (\text{object } \beta)$ 
and  $bn \ \alpha \#* Q$ 

```

```

shows  $\text{distinct}(bn \ \beta)$ 
using assms
proof -
  {
    fix  $xvec \ M \ yvec \ N$ 
    assume Eq:  $(\nu*xvec)N \prec' P = (\nu*yvec)M \prec' Q$ 
    assume  $\text{distinct } xvec$  and  $xvec \#* M$  and  $xvec \#* yvec$  and  $xvec \#* Q$ 
    assume Mem:  $\bigwedge x. x \in \text{set } xvec \implies x \in (\text{supp } N)$ 
    have  $\text{distinct } yvec$ 
    proof -
      from Eq have  $\text{length } xvec = \text{length } yvec$ 
        by(rule boundOutputChainEqLength)
      with Eq  $\langle \text{distinct } xvec \rangle \langle xvec \#* yvec \rangle \langle xvec \#* M \rangle \langle xvec \#* Q \rangle$  Mem show
?thesis
    proof(induct n==length xvec arbitrary: xvec yvec M Q rule: nat.induct)
      case(zero xvec yvec M Q)
      then show ?case by simp
    next
      case(Suc n xvec yvec M Q)
      have L:  $\text{length } xvec = \text{length } yvec$  and  $\text{Suc } n = \text{length } xvec$  by fact+

```

then obtain $x \text{ xvec}' y \text{ yvec}'$ **where** $xEq: xvec = x\#\text{xvec}'$ **and** $yEq: yvec = y\#\text{yvec}'$
and $L': \text{length } xvec' = \text{length } yvec'$
by(*cases xvec, auto, cases yvec, auto*)
have $xvecFreshyvec: xvec \#* yvec$ **and** $xvecDist: \text{distinct } xvec$ **by fact**
with $xEq yEq$ **have** $xineqy: x \neq y$ **and** $xvec'Freshyvec': xvec' \#* yvec'$
and $xvec'Dist: \text{distinct } xvec'$ **and** $xFreshxvec': x \# xvec'$
and $xFreshyvec': x \# yvec'$ **and** $yFreshxvec': y \# xvec'$
by auto
have $Eq: (\nu*xvec)N \prec' P = (\nu*yvec)M \prec' Q$ **by fact**
with $xEq yEq xineqy$ **have** $Eq': (\nu*xvec')N \prec' P = (\nu*([(x, y)] \cdot yvec'))([(x, y)] \cdot M) \prec' ([x, y] \cdot Q)$
by(*simp add: boundOutput.inject alpha eqvts*)
moreover have $Mem: \bigwedge x. x \in \text{set } xvec \implies x \in \text{supp } N$ **by fact**
with xEq **have** $\bigwedge x. x \in \text{set } xvec' \implies x \in \text{supp } N$ **by simp**
moreover have $xvec \#* M$ **by fact**
with $xEq xFreshxvec' yFreshxvec'$ **have** $xvec' \#* ([x, y] \cdot M)$ **by simp**
moreover have $xvecFreshQ: xvec \#* Q$ **by fact**
with $xEq xFreshxvec' yFreshxvec'$ **have** $xvec' \#* ([x, y] \cdot Q)$ **by simp**
moreover have $Suc \ n = \text{length } xvec$ **by fact**
with xEq **have** $n = \text{length } xvec'$ **by simp**
moreover from $xvec'Freshyvec' xFreshxvec' yFreshxvec'$ **have** $xvec' \#* ([x, y] \cdot yvec')$
by simp
moreover from L' **have** $\text{length } xvec' = \text{length}([x, y] \cdot yvec')$ **by simp**
ultimately have $\text{distinct}([x, y] \cdot yvec')$ **using** $xvec'Dist$
apply –
apply(*rule Suc*)
by(*assumption | simp*)
then have $\text{distinct } yvec'$ **by simp**
from $Mem \ xEq$ **have** $xSuppN: x \in \text{supp } N$ **by simp**
from $L \langle \text{distinct } xvec \rangle \langle xvec \#* yvec \rangle \langle xvec \#* M \rangle \langle xvec \#* Q \rangle$
have $(\nu*yvec)M \prec' Q = (\nu*xvec)([yvec \ xvec] \cdot_v M) \prec' ([yvec \ xvec] \cdot_v Q)$
by(*simp add: alphaBoundOutputChain''*)
with Eq **have** $N = [yvec \ xvec] \cdot_v M$ **by simp**
with $xEq yEq$ **have** $N = [(y, x)] \cdot [yvec' \ xvec'] \cdot_v M$
by simp
with $xSuppN$ **have** $ySuppM: y \in \text{supp}([yvec' \ xvec'] \cdot_v M)$
by(*force simp add: calc-atm eqvts name-swap*)
dest: pt-set-bij2[where pi=[(x, y)], OF pt-name-inst, OF at-name-inst]
have $y \# yvec'$
proof –
{
assume $y \in \text{supp } yvec'$
then have $y \in \text{set } yvec'$
by(*induct yvec'*) (*auto simp add: supp-list-nil supp-list-cons supp-atm*)
moreover from $\langle xvec \#* M \rangle \ xEq \ xFreshxvec'$ **have** $xvec' \#* M$ **by simp**
ultimately have $y \# [yvec' \ xvec'] \cdot_v M$ **using** $L' \ xvec'Freshyvec' \ xvec'Dist$
by(*force intro: freshChainPerm*)

```

    with  $ySuppM$  have False by(simp add: fresh-def)
  }
  then show ?thesis
    by(simp add: fresh-def, rule notI)
  qed
  with  $\langle distinct\ yvec' \rangle\ yEq$  show ?case by simp
  qed
  qed
} note res = this
show ?thesis
  apply(rule actionCases[where  $\alpha=\alpha$ ])
  using assms res
  by(auto simp add: residualInject supp-some)
qed

lemma boundOutputDistinct:
  fixes  $\Psi$     :: 'b
  and  $P$       :: ('a, 'b, 'c) psi
  and  $\alpha$     :: 'a action
  and  $P'$     :: ('a, 'b, 'c) psi

assumes  $\Psi \triangleright P \mapsto \alpha \prec P'$ 

shows distinct(bn  $\alpha$ )
  using assms
proof(nominal-induct  $\Psi P x\exists==\alpha \prec P'$  avoiding:  $\alpha P'$  rule: semantics.strong-induct)
  case cPar1
  then show ?case
    by(force intro: alphaDistinct boundOutputBindObject)
  next
  case cPar2
  then show ?case
    by(force intro: alphaDistinct boundOutputBindObject)
  next
  case (cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q \alpha P''$ )
  note  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{jM}(\nu*xvec)\langle N \rangle \prec Q' \rangle$ 
  moreover from  $\langle xvec \#* M \rangle$  have  $bn(\text{jM}(\nu*xvec)\langle N \rangle) \#* subject(\text{jM}(\nu*xvec)\langle N \rangle)$ 
  by simp
  moreover from  $\langle distinct\ xvec \rangle$  have distinct (bn (jM( $\nu*xvec$ ) $\langle N \rangle$ )) by simp
  ultimately have someX:  $\bigwedge x. x \in set(bn(\text{jM}(\nu*xvec)\langle N \rangle)) \implies x \in supp$ 
  (object(jM( $\nu*xvec$ ) $\langle N \rangle$ ))
  by (drule boundOutputBindObject) (assumption)

  note  $\langle \text{jM}(\nu*xvec)\langle N \rangle \prec P' \parallel Q' = \alpha \prec P'' \rangle$ 
  moreover from distinct xvec have distinct (bn (jM( $\nu*xvec$ ) $\langle N \rangle$ )) by simp
  moreover note  $\langle \bigwedge x. x \in set(bn(\text{jM}(\nu*xvec)\langle N \rangle)) \implies x \in supp(object$ 
  ( $\text{jM}(\nu*xvec)\langle N \rangle$ ) $\rangle$ 
  moreover from  $\langle xvec \#* \alpha \rangle$  have  $bn(\text{jM}(\nu*xvec)\langle N \rangle) \#* bn\ \alpha$  by simp
  moreover from  $\langle xvec \#* \alpha \rangle$  have  $bn(\text{jM}(\nu*xvec)\langle N \rangle) \#* object\ \alpha$  by simp

```

```

moreover from ⟨xvec #* P''⟩ have bn (jM(ν*xvec)⟨N⟩) #* P'' by simp
ultimately show ?case
  by(rule alphaDistinct)
next
case (cBrComm2 Ψ ΨQ P M xvec N P' AP ΨP Q Q' AQ α P'')
note ⟨Ψ ⊗ ΨQ ▷ P ⟶ jM(ν*xvec)⟨N⟩ < P'⟩
moreover from ⟨xvec #* M⟩ have bn (jM(ν*xvec)⟨N⟩) #* subject (jM(ν*xvec)⟨N⟩)
by simp
moreover from ⟨distinct xvec⟩ have distinct (bn (jM(ν*xvec)⟨N⟩)) by simp
ultimately have someX: ∧ x. x ∈ set (bn (jM(ν*xvec)⟨N⟩)) ⟹ x ∈ supp
(object (jM(ν*xvec)⟨N⟩))
  by (drule boundOutputBindObject) (assumption)

note ⟨jM(ν*xvec)⟨N⟩ < P' || Q' = α < P''⟩
moreover from ⟨distinct xvec⟩ have distinct (bn (jM(ν*xvec)⟨N⟩)) by simp
moreover note ⟨∧x. x ∈ set (bn (jM(ν*xvec)⟨N⟩)) ⟹ x ∈ supp (object
(jM(ν*xvec)⟨N⟩))⟩
moreover from ⟨xvec #* α⟩ have bn (jM(ν*xvec)⟨N⟩) #* bn α by simp
moreover from ⟨xvec #* α⟩ have bn (jM(ν*xvec)⟨N⟩) #* object α by simp
moreover from ⟨xvec #* P''⟩ have bn (jM(ν*xvec)⟨N⟩) #* P'' by simp
ultimately show ?case
  by(rule alphaDistinct)
next
case(cBrClose Ψ P M xvec N P' α P'')
then show ?case by(simp add: residualInject)
next
case(cOpen Ψ P M xvec yvec N P' x α P'')
note ⟨M(ν*(xvec@x#yvec))⟨N⟩ < P' = α < P''⟩
moreover from ⟨xvec #* yvec⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩
have distinct(bn(M(ν*(xvec@x#yvec))⟨N⟩))
  by auto (simp add: fresh-star-def fresh-def name-list-supp)
moreover {
  fix y
  from ⟨Ψ ▷ P ⟶ M(ν*(xvec@yvec))⟨N⟩ < P'⟩ ⟨x ∈ supp N⟩ ⟨x # xvec⟩ ⟨x #
yvec⟩ ⟨x # M⟩ ⟨x # Ψ⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩ ⟨xvec #* Ψ⟩ ⟨xvec #* P⟩ ⟨xvec
#* M⟩ ⟨xvec #* yvec⟩ ⟨yvec #* Ψ⟩ ⟨yvec #* P⟩ ⟨yvec #* M⟩
  have Ψ ▷ (νx)P ⟶ M(ν*(xvec@x#yvec))⟨N⟩ < P' by(rule semantics.cOpen)
  moreover moreover from ⟨xvec #* M⟩ ⟨x # M⟩ ⟨yvec #* M⟩
  have bn(M(ν*(xvec@x#yvec))⟨N⟩) #* (subject(M(ν*(xvec@x#yvec))⟨N⟩))
    by simp
  moreover note ⟨distinct(bn(M(ν*(xvec@x#yvec))⟨N⟩))⟩
  moreover assume y ∈ set(bn(M(ν*(xvec@x#yvec))⟨N⟩))

  ultimately have y ∈ supp(object(M(ν*(xvec@x#yvec))⟨N⟩))
    by(metis boundOutputBindObject)
}
moreover from ⟨xvec #* α⟩ ⟨x # α⟩ ⟨yvec #* α⟩
have bn(M(ν*(xvec@x#yvec))⟨N⟩) #* bn α and bn(M(ν*(xvec@x#yvec))⟨N⟩)
#* object α by simp+

```

```

moreover from ⟨xvec #* P''⟩ ⟨x # P''⟩ ⟨yvec #* P''⟩
have  $bn(M(\nu^*(xvec@x\#yvec))\langle N \rangle) \#* P''$  by simp
ultimately show ?case by(rule alphaDistinct)
next
case(cBrOpen  $\Psi$  P M xvec yvec N P' x  $\alpha$  P'')
note ⟨ $iM(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P' = \alpha \prec P''$ ⟩
moreover from ⟨xvec #* yvec⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩
have distinct( $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle)$ )
  by auto (simp add: fresh-star-def fresh-def name-list-supp)
moreover {
  fix y
  from ⟨ $\Psi \triangleright P \mapsto iM(\nu^*(xvec@yvec))\langle N \rangle \prec P'$ ⟩ ⟨x ∈ supp N⟩ ⟨x # xvec⟩ ⟨x # yvec⟩
  ⟨x # M⟩ ⟨x #  $\Psi$ ⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩ ⟨xvec #*  $\Psi$ ⟩ ⟨xvec #* P⟩ ⟨xvec #* M⟩
  ⟨xvec #* yvec⟩ ⟨yvec #*  $\Psi$ ⟩ ⟨yvec #* P⟩ ⟨yvec #* M⟩
  have  $\Psi \triangleright (\nu x)P \mapsto iM(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P'$  by(rule semantics.cBrOpen)
  moreover moreover from ⟨xvec #* M⟩ ⟨x # M⟩ ⟨yvec #* M⟩
  have  $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle) \#* (subject(iM(\nu^*(xvec@x\#yvec))\langle N \rangle))$ 
    by simp
  moreover note ⟨distinct( $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle)$ )⟩
  moreover assume y ∈ set( $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle)$ )

  ultimately have y ∈ supp( $object(iM(\nu^*(xvec@x\#yvec))\langle N \rangle)$ )
    by(metis boundOutputBindObject)
}
moreover from ⟨xvec #*  $\alpha$ ⟩ ⟨x #  $\alpha$ ⟩ ⟨yvec #*  $\alpha$ ⟩
have  $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle) \#* bn \alpha$  and  $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle)$ 
#* object  $\alpha$  by simp+
moreover from ⟨xvec #* P''⟩ ⟨x # P''⟩ ⟨yvec #* P''⟩
have  $bn(iM(\nu^*(xvec@x\#yvec))\langle N \rangle) \#* P''$  by simp
ultimately show ?case by(rule alphaDistinct)
next
case cScope
then show ?case
  by – (rule alphaDistinct, auto intro: boundOutputBindObject)
qed (simp-all add: residualInject)

lemma inputDistinct:
  fixes  $\Psi$  :: 'b
  and M :: 'a
  and xvec :: name list
  and N :: 'a
  and P :: ('a, 'b, 'c) psi
  and Rs :: ('a, 'b, 'c) residual

assumes  $\Psi \triangleright M(\lambda^*xvec N).P \mapsto Rs$ 

shows distinct xvec
using assms
by(nominal-induct  $\Psi P == M(\lambda^*xvec N).P Rs$  avoiding: xvec N P rule: seman-)

```


tics.strong-induct)

(*auto simp add: psi.inject intro: alphaInputDistinct*)

lemma *outputInduct* [*consumes 2, case-names cAlpha cOutput cCase cPar1 cPar2 cOpen cScope cBang*]:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and *yvec* :: name list
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and *Prop* :: 'f::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow
'a \Rightarrow name list \Rightarrow 'a \Rightarrow ('a, 'b, 'c) psi \Rightarrow bool
and C :: 'f::fs-name

assumes $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$

and $xvec \#* M$

and *rAlpha*: $\bigwedge \Psi P M xvec N P' p C. \llbracket xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; xvec \#* (p \cdot xvec);$

$set\ p \subseteq set\ xvec \times set(p \cdot xvec); distinctPerm\ p;$
 $(p \cdot xvec) \#* N; (p \cdot xvec) \#* P'; Prop\ C\ \Psi\ P\ M$

$xvec\ N\ P' \rrbracket \Longrightarrow$

$Prop\ C\ \Psi\ P\ M\ (p \cdot xvec)\ (p \cdot N)\ (p \cdot P')$

and *rOutput*: $\bigwedge \Psi M K N P C. \llbracket \Psi \vdash M \leftrightarrow K \rrbracket \Longrightarrow Prop\ C\ \Psi\ (M\langle N \rangle.P)\ K\ (\llbracket N\ P$

and *rCase*: $\bigwedge \Psi P M xvec N P' \varphi Cs C. \llbracket \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. Prop\ C\ \Psi\ P\ M\ xvec\ N\ P'; (\varphi, P) \in set\ Cs; \Psi \vdash \varphi; guarded\ P \rrbracket \Longrightarrow$

$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ xvec\ N\ P'$

and *rPar1*: $\bigwedge \Psi \Psi_Q P M xvec N P' A_Q Q C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; extractFrame\ Q = \langle A_Q, \Psi_Q \rangle; distinct\ A_Q;$

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ xvec\ N\ P';$

$A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M;$

$A_Q \#* xvec; A_Q \#* N; A_Q \#* P'; A_Q \#* C; xvec \#* Q;$

$xvec \#* \Psi; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#* C \rrbracket \Longrightarrow$

$Prop\ C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P' \parallel Q)$

and *rPar2*: $\bigwedge \Psi \Psi_P Q M xvec N Q' A_P P C.$

$\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu*xvec)\langle N \rangle \prec Q'; extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P;$

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ xvec\ N\ Q';$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M;$

$A_P \#* xvec; A_P \#* N; A_P \#* Q'; A_P \#* C; xvec \#* Q;$

$xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* M; xvec \#* C \rrbracket \Longrightarrow$

$Prop\ C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P \parallel Q')$

and *rOpen*: $\bigwedge \Psi P M xvec yvec N P' x C.$

$\llbracket \Psi \triangleright P \mapsto M(\nu*(xvec@yvec))\langle N \rangle \prec P'; x \in supp\ N; \bigwedge C. Prop\ C\ \Psi\ P\ M\ (xvec@yvec)\ N\ P';$

$x \#* \Psi; x \#* M; x \#* xvec; x \#* yvec; xvec \#* \Psi; xvec \#* P; xvec \#* M;$

$yvec \#* \Psi; yvec \#* P; yvec \#* M; yvec \#* C; x \#* C; xvec \#* C \rrbracket \Longrightarrow$

$$\text{and } rScope: \bigwedge \Psi P M \text{ xvec } N P' x C. \\
\llbracket \Psi \triangleright P \mapsto M(\nu * \text{xvec}) \langle N \rangle \prec P'; \bigwedge C. \text{ Prop } C \Psi P M \text{ xvec } N P'; \\
x \# \Psi; x \# M; x \# \text{xvec}; x \# N; \text{xvec} \#* \Psi; \\
\text{xvec} \#* P; \text{xvec} \#* M; x \# C; \text{xvec} \#* C \rrbracket \implies \\
\text{Prop } C \Psi (\nu x) P M \text{ xvec } N (\nu x) P'$$

$$\text{and } rBang: \bigwedge \Psi P M \text{ xvec } N P' C. \\
\llbracket \Psi \triangleright P \parallel !P \mapsto M(\nu * \text{xvec}) \langle N \rangle \prec P'; \text{ guarded } P; \bigwedge C. \text{ Prop } C \\
\Psi (P \parallel !P) M \text{ xvec } N P' \rrbracket \implies \\
\text{Prop } C \Psi (!P) M \text{ xvec } N P'$$

shows $\text{Prop } C \Psi P M \text{ xvec } N P'$

proof –

note $\langle \Psi \triangleright P \mapsto M(\nu * \text{xvec}) \langle N \rangle \prec P' \rangle$

moreover from $\langle \text{xvec} \#* M \rangle$ **have** $\text{bn}(M(\nu * \text{xvec}) \langle N \rangle) \#* \text{subject}(M(\nu * \text{xvec}) \langle N \rangle)$

by *simp*

moreover from $\langle \Psi \triangleright P \mapsto M(\nu * \text{xvec}) \langle N \rangle \prec P' \rangle$ **have** $\text{distinct}(\text{bn}(M(\nu * \text{xvec}) \langle N \rangle))$

by(*rule boundOutputDistinct*)

ultimately show *?thesis*

proof(*nominal-induct* $\Psi P \alpha = M(\nu * \text{xvec}) \langle N \rangle P'$ *avoiding: C arbitrary: M xvec N rule: semanticsInduct*)

case(*cAlpha* $\Psi P \alpha P' p C M \text{ xvec } N$)

from $\langle (p \cdot \alpha) = M(\nu * \text{xvec}) \langle N \rangle \rangle$ **have** $\langle p \cdot p \cdot \alpha \rangle = p \cdot (M(\nu * \text{xvec}) \langle N \rangle)$

by(*simp add: fresh-bij*)

with $\langle \text{distinctPerm } p \rangle$ **have** $A: \alpha = (p \cdot M)(\nu * (p \cdot \text{xvec})) \langle (p \cdot N) \rangle$

by(*simp add: eqts*)

with $\langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle \langle \text{bn } \alpha \#* \text{bn}(p \cdot \alpha) \rangle \langle \text{distinctPerm } p \rangle$

have $\langle (p \cdot \text{xvec}) \#* \Psi \text{ and } (p \cdot \text{xvec}) \#* P \text{ and } (p \cdot \text{xvec}) \#* (p \cdot M) \text{ and } (p \cdot \text{xvec}) \#* C \text{ and } (p \cdot \text{xvec}) \#* (p \cdot p \cdot \text{xvec}) \rangle$

by *auto*

moreover from $A \langle \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha)) \rangle \langle \text{distinctPerm } p \rangle$

have $S: \text{set } p \subseteq \text{set}(p \cdot \text{xvec}) \times \text{set}(p \cdot p \cdot \text{xvec})$ **by** *simp*

moreover note $\langle \text{distinctPerm } p \rangle$

moreover from $A \langle \text{bn}(p \cdot \alpha) \#* \alpha \rangle \langle \text{bn}(p \cdot \alpha) \#* P' \rangle$

have $\langle (p \cdot p \cdot \text{xvec}) \#* (p \cdot N) \text{ and } (p \cdot p \cdot \text{xvec}) \#* P' \text{ by } \text{simp} +$

moreover from A **have** $\text{Prop } C \Psi P (p \cdot M) (p \cdot \text{xvec}) (p \cdot N) P'$

by(*rule cAlpha*)

ultimately have $\text{Prop } C \Psi P (p \cdot M) (p \cdot p \cdot \text{xvec}) (p \cdot p \cdot N) (p \cdot P')$

by(*rule rAlpha*)

moreover from $A \langle \text{bn } \alpha \#* \text{subject } \alpha \rangle$ **have** $\langle (p \cdot \text{xvec}) \#* (p \cdot M) \text{ by } \text{simp}$

then have $\text{xvec} \#* M$ **by**(*simp add: fresh-star-bij*)

from $A \langle \text{bn}(p \cdot \alpha) \#* \alpha \rangle \langle \text{distinctPerm } p \rangle$ **have** $\text{xvec} \#* (p \cdot M)$ **by** *simp*

then have $\langle (p \cdot \text{xvec}) \#* (p \cdot p \cdot M) \text{ by } (\text{simp add: fresh-star-bij})$

with $\langle \text{distinctPerm } p \rangle$ **have** $\langle (p \cdot \text{xvec}) \#* M \text{ by } \text{simp}$

with $\langle \text{xvec} \#* M \rangle S \langle \text{distinctPerm } p \rangle$ **have** $\langle (p \cdot M) = M \text{ by } \text{simp}$

ultimately show *?case using S* $\langle \text{distinctPerm } p \rangle$ **by** *simp*

next

case *cInput*

then show *?case by*(*simp add: residualInject*)

```

next
  case cBrInput
  then show ?case by simp
next
  case cOutput
  then show ?case by(force dest: rOutput simp add: action.inject)
next
  case cBrOutput
  then show ?case by simp
next
  case cCase
  then show ?case by(force intro: rCase)
next
  case cPar1
  then show ?case by(force intro: rPar1)
next
  case cPar2
  then show ?case by(force intro: rPar2)
next
  case cComm1
  then show ?case by(simp add: action.inject)
next
  case cComm2
  then show ?case by(simp add: action.inject)
next
  case cBrMerge
  then show ?case by(simp add: action.inject)
next
  case cBrComm1
  then show ?case by simp
next
  case cBrComm2
  then show ?case by simp
next
  case cBrClose
  then show ?case by simp
next
  case cOpen
  then show ?case by(auto intro: rOpen simp add: action.inject)
next
  case cBrOpen
  then show ?case by simp
next
  case cScope
  then show ?case by(auto intro: rScope)
next
  case cBang
  then show ?case by(auto intro: rBang)
qed

```

qed

lemma *brOutputInduct'*[*consumes 2, case-names cAlpha cBrOutput cCase cPar1 cPar2 cBrComm1 cBrComm2 cBrOpen cScope cBang*]:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and $yvec$:: name list
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $Prop$:: 'f::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow
'a \Rightarrow name list \Rightarrow 'a \Rightarrow ('a, 'b, 'c) psi \Rightarrow bool
and C :: 'f::fs-name

assumes $\Psi \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'$

and $xvec \#* M$

and $rAlpha$: $\bigwedge \Psi P M xvec N P' p C. \llbracket xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; xvec \#* (p \cdot xvec) \rrbracket$

set $p \subseteq \text{set } xvec \times \text{set}(p \cdot xvec)$; *distinctPerm* p ;
 $(p \cdot xvec) \#* N$; $(p \cdot xvec) \#* P'$; *Prop* $C \Psi P M$

$xvec N P' \rrbracket \Longrightarrow$

Prop $C \Psi P M (p \cdot xvec) (p \cdot N) (p \cdot P')$

and $rBrOutput$: $\bigwedge \Psi M K N P C. \llbracket \Psi \vdash M \preceq K \rrbracket \Longrightarrow \text{Prop } C \Psi (M\langle N \rangle.P) K$
 $(\parallel) N P$

and $rCase$: $\bigwedge \Psi P M xvec N P' \varphi Cs C. \llbracket \Psi \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. \text{Prop } C \Psi P M xvec N P'; (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P \rrbracket \Longrightarrow$

Prop $C \Psi (Cases Cs) M xvec N P'$

and $rPar1$: $\bigwedge \Psi \Psi_Q P M xvec N P' A_Q Q C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$

$\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P M xvec N P';$

$A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M;$

$A_Q \#* xvec; A_Q \#* N; A_Q \#* P'; A_Q \#* C; xvec \#* Q;$

$xvec \#* \Psi; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#* C \rrbracket \Longrightarrow$

Prop $C \Psi (P \parallel Q) M xvec N (P' \parallel Q)$

and $rPar2$: $\bigwedge \Psi \Psi_P Q M xvec N Q' A_P P C.$

$\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$

$\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q M xvec N Q';$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M;$

$A_P \#* xvec; A_P \#* N; A_P \#* Q'; A_P \#* C; xvec \#* Q;$

$xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* M; xvec \#* C \rrbracket \Longrightarrow$

Prop $C \Psi (P \parallel Q) M xvec N (P \parallel Q')$

and $rBrComm1$: $\bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M\langle N \rangle \prec P';$

$\text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$

$\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'; \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P)$

$Q M xvec N Q';$

$\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$

$A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; \text{distinct } xvec;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C \rceil \implies$
 $\text{Prop } C \Psi (P \parallel Q) M \text{vec } N (P' \parallel Q')$
and $rBrComm2: \bigwedge \Psi \Psi_Q P M \text{vec } N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'; \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q)$
 $P M \text{vec } N P';$
 $\text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M \langle N \rangle \prec Q';$
 $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; \text{distinct } xvec;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C \rceil \implies$
 $\text{Prop } C \Psi (P \parallel Q) M \text{vec } N (P' \parallel Q')$
and $rOpen: \bigwedge \Psi P M \text{vec } yvec N P' x C.$
 $\llbracket \Psi \triangleright P \mapsto_i M(\nu * (xvec @ yvec)) \langle N \rangle \prec P'; x \in \text{supp } N; \bigwedge C. \text{Prop}$
 $C \Psi P M (xvec @ yvec) N P';$
 $x \# \Psi; x \# M; x \# xvec; x \# yvec; xvec \# \Psi; xvec \# P; xvec \# M;$
 $yvec \# \Psi; yvec \# P; yvec \# M; yvec \# C; x \# C; xvec \# C \rceil \implies$
 $\text{Prop } C \Psi ((\nu x)P) M (xvec @ x \# yvec) N P'$
and $rScope: \bigwedge \Psi P M \text{vec } N P' x C.$
 $\llbracket \Psi \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'; \bigwedge C. \text{Prop } C \Psi P M \text{vec } N P';$
 $x \# \Psi; x \# M; x \# xvec; x \# N; xvec \# \Psi;$
 $xvec \# P; xvec \# M; x \# C; xvec \# C \rceil \implies$
 $\text{Prop } C \Psi ((\nu x)P) M \text{vec } N ((\nu x)P')$
and $rBang: \bigwedge \Psi P M \text{vec } N P' C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'; \text{guarded } P; \bigwedge C. \text{Prop } C$
 $\Psi (P \parallel !P) M \text{vec } N P' \rceil \implies$
 $\text{Prop } C \Psi (!P) M \text{vec } N P'$
shows $\text{Prop } C \Psi P M \text{vec } N P'$
proof –
note $\langle \Psi \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P' \rangle$
moreover from $\langle xvec \#* M \rangle$ **have** $bn(iM(\nu * xvec) \langle N \rangle) \#* \text{subject}(iM(\nu * xvec) \langle N \rangle)$
by simp
moreover from $\langle \Psi \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P' \rangle$ **have** $\text{distinct}(bn(iM(\nu * xvec) \langle N \rangle))$
by $(\text{rule boundOutputDistinct})$
ultimately show $?thesis$
proof $(\text{nominal-induct } \Psi P \alpha = iM(\nu * xvec) \langle N \rangle P' \text{ avoiding: } C \text{ arbitrary: } M \text{vec}$
 $N \text{ rule: semanticsInduct})$
case $(cAlpha \Psi P \alpha P' p C M \text{vec } N)$
from $\langle (p \cdot \alpha) = iM(\nu * xvec) \langle N \rangle \rangle$ **have** $(p \cdot p \cdot \alpha) = p \cdot (iM(\nu * xvec) \langle N \rangle)$
by $(\text{simp add: fresh-bij})$
with $\langle \text{distinctPerm } p \rangle$ **have** $A: \alpha = i(p \cdot M)(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle$

```

    by(simp add: eqts)
    with ⟨bn α #* Ψ⟩ ⟨bn α #* P⟩ ⟨bn α #* subject α⟩ ⟨bn α #* C⟩ ⟨bn α #* bn(p
    · α)⟩ ⟨distinctPerm p⟩
    have (p · xvec) #* Ψ and (p · xvec) #* P and (p · xvec) #* (p · M) and (p
    · xvec) #* C and (p · xvec) #* (p · p · xvec)
    by auto
    moreover from A ⟨set p ⊆ set(bn α) × set(bn(p · α))⟩ ⟨distinctPerm p⟩
    have S: set p ⊆ set(p · xvec) × set(p · p · xvec) by simp
    moreover note ⟨distinctPerm p⟩
    moreover from A ⟨bn(p · α) #* α⟩ ⟨bn(p · α) #* P'⟩
    have (p · p · xvec) #* (p · N) and (p · p · xvec) #* P' by simp+
    moreover from A have Prop C Ψ P (p · M) (p · xvec) (p · N) P'
    by(rule cAlpha)
    ultimately have Prop C Ψ P (p · M) (p · p · xvec) (p · p · N) (p · P')
    by(rule rAlpha)
    moreover from A ⟨bn α #* subject α⟩ have (p · xvec) #* (p · M) by simp
    then have xvec #* M by(simp add: fresh-star-bij)
    from A ⟨bn(p · α) #* α⟩ ⟨distinctPerm p⟩ have xvec #* (p · M) by simp
    then have (p · xvec) #* (p · p · M) by(simp add: fresh-star-bij)
    with ⟨distinctPerm p⟩ have (p · xvec) #* M by simp
    with ⟨xvec #* M⟩ S ⟨distinctPerm p⟩ have (p · M) = M by simp
    ultimately show ?case using S ⟨distinctPerm p⟩ by simp
next
  case cInput
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by simp
next
  case cOutput
  then show ?case by simp
next
  case cBrOutput
  then show ?case by(simp add: rBrOutput action.inject)
next
  case cCase
  then show ?case by(force intro: rCase)
next
  case cPar1
  then show ?case by(force intro: rPar1)
next
  case cPar2
  then show ?case by(force intro: rPar2)
next
  case cComm1
  then show ?case by(simp add: action.inject)
next
  case cComm2
  then show ?case by(simp add: action.inject)

```

```

next
  case cBrMerge
  then show ?case by(simp add: action.inject)
next
  case cBrComm1
  then show ?case
  by(auto intro: rBrComm1 simp add: action.inject)
next
  case cBrComm2
  then show ?case
  by(auto intro: rBrComm2 simp add: action.inject)
next
  case cBrClose
  then show ?case by simp
next
  case cOpen
  then show ?case by simp
next
  case cBrOpen
  then show ?case by(auto intro: rOpen simp add: action.inject)
next
  case cScope
  then show ?case by(auto intro: rScope)
next
  case cBang
  then show ?case by(auto intro: rBang)
qed
qed

```

lemma *inputInduct*[*consumes 1*, *case-names cInput cCase cPar1 cPar2 cScope cBang*]:

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $Prop$  :: 'f::fs-name  $\Rightarrow$  'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$ 
           'a  $\Rightarrow$  'a  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$  bool
and  $C$      :: 'f::fs-name

```

assumes *Trans*: $\Psi \triangleright P \mapsto M(\downarrow N) \prec P'$

```

and rInput:  $\bigwedge \Psi M K \text{ xvec } N \text{ Tvec } P C.$ 
            $\llbracket \Psi \vdash M \leftrightarrow K; \text{ distinct } \text{xvec}; \text{ set } \text{xvec} \subseteq \text{supp } N;$ 
            $\text{ length } \text{xvec} = \text{ length } \text{Tvec}; \text{ xvec } \#* \Psi;$ 
            $\text{ xvec } \#* M; \text{ xvec } \#* K; \text{ xvec } \#* C \rrbracket \implies$ 
            $\text{ Prop } C \Psi (M(\lambda * \text{xvec } N).P)$ 
            $K (N[\text{xvec} ::= \text{Tvec}]) (P[\text{xvec} ::= \text{Tvec}])$ 

```

and *rCase*: $\bigwedge \Psi P M N P' \varphi Cs C. \llbracket \Psi \triangleright P \mapsto M(\downarrow N) \prec P'; \bigwedge C. \text{ Prop } C \Psi$
 $P M N P'; (\varphi, P) \in \text{ set } Cs; \Psi \vdash \varphi; \text{ guarded } P \rrbracket \implies$

$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ N\ P'$

and $rPar1: \bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_Q\ Q\ C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$
distinct $A_Q;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ N\ P'; distinct\ A_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* N;$
 $A_Q \#* P'; A_Q \#* C \rrbracket \implies$
 $Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P' \parallel Q)$

and $rPar2: \bigwedge \Psi\ \Psi_P\ Q\ M\ N\ Q'\ A_P\ P\ C.$
 $\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(N) \prec Q'; extractFrame\ P = \langle A_P, \Psi_P \rangle;$
distinct $A_P;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ N\ Q'; distinct\ A_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* N;$
 $A_P \#* Q'; A_P \#* C \rrbracket \implies$
 $Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P \parallel Q')$

and $rScope: \bigwedge \Psi\ P\ M\ N\ P'\ x\ C.$
 $\llbracket \Psi \triangleright P \mapsto M(N) \prec P'; \bigwedge C. Prop\ C\ \Psi\ P\ M\ N\ P'; x \# \Psi; x \#$
 $M; x \# N; x \# C \rrbracket \implies$
 $Prop\ C\ \Psi\ ((\nu x)P)\ M\ N\ ((\nu x)P')$

and $rBang: \bigwedge \Psi\ P\ M\ N\ P'\ C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto M(N) \prec P'; guarded\ P; \bigwedge C. Prop\ C\ \Psi\ (P \parallel$
 $!P)\ M\ N\ P' \rrbracket \implies Prop\ C\ \Psi\ (!P)\ M\ N\ P'$

shows $Prop\ C\ \Psi\ P\ M\ N\ P'$
using *Trans*
proof(*nominal-induct* $\Psi\ P\ Rs == M(N) \prec P'$ *avoiding: C arbitrary: P' rule: se-*
mantics.strong-induct)
case(*cInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$)
then show *?case*
by(*force intro: rInput simp add: residualInject action.inject*)
next
case(*cBrInput* $\Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$)
then show *?case*
by (*simp add: residualInject*)
next
case(*Output* $\Psi\ M\ K\ N\ P\ C$)
then show *?case by(simp add: residualInject)*
next
case *BrOutput*
then show *?case by(simp add: residualInject)*
next
case(*Case* $\Psi\ P\ \varphi\ CS\ C\ P'$)
then show *?case by(force intro: rCase)*
next
case(*cPar1* $\Psi\ \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q\ C\ P''$)
then show *?case by(force intro: rPar1 simp add: residualInject)*
next
case(*cPar2* $\Psi\ \Psi_P\ Q\ \alpha\ Q'\ xvec\ P\ C\ Q''$)
then show *?case by(force intro: rPar2 simp add: residualInject)*
next


```

    case(cComm1  $\Psi \Psi Q P M N P' xvec \Psi P Q K yvec Q' yvec C PQ$ )
    then show ?case by(simp add: residualInject)
next
    case(cComm2  $\Psi \Psi Q P M zvec N P' xvec \Psi P Q K yvec Q' C PQ$ )
    then show ?case by(simp add: residualInject)
next
    case cBrMerge
    then show ?case by(simp add: residualInject)
next
    case cBrComm1
    then show ?case by(simp add: residualInject)
next
    case cBrComm2
    then show ?case by(simp add: residualInject)
next
    case(cBrClose  $\Psi P M xvec N P' C P''$ )
    then show ?case by(simp add: residualInject)
next
    case(cOpen  $\Psi P M xvec N P' x yvec C P''$ )
    then show ?case by(simp add: residualInject)
next
    case(cBrOpen  $\Psi P M xvec N P' x yvec C P''$ )
    then show ?case by(simp add: residualInject)
next
    case(cScope  $\Psi P \alpha P' x C P''$ )
    then show ?case by(force intro: rScope simp add: residualInject)
next
    case(Bang  $\Psi P C P'$ )
    then show ?case by(force intro: rBang)
qed

```

lemma *brInputInduct*[consumes 1, case-names *cBrInput cCase cPar1 cPar2 cBrMerge cScope cBang*]:

```

fixes  $\Psi$     :: 'b
and  $P$       :: ('a, 'b, 'c) psi
and  $M$       :: 'a
and  $N$       :: 'a
and  $P'$      :: ('a, 'b, 'c) psi
and  $Prop$    :: 'f::fs-name  $\Rightarrow$  'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$ 
              'a  $\Rightarrow$  'a  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$  bool
and  $C$       :: 'f::fs-name

```

assumes *Trans*: $\Psi \triangleright P \mapsto_i M(\downarrow N) \prec P'$

```

and rBrInput:  $\bigwedge \Psi K M xvec N Tvec P C.$ 
     $\llbracket \Psi \vdash K \succeq M; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N;$ 
     $\text{length } xvec = \text{length } Tvec; xvec \#* \Psi;$ 
     $xvec \#* M; xvec \#* K; xvec \#* C \rrbracket \Longrightarrow$ 
     $Prop C \Psi (M(\downarrow \lambda xvec N).P)$ 
     $K (N[xvec ::= Tvec]) (P[xvec ::= Tvec])$ 

```

and $rCase: \bigwedge \Psi P M N P' \varphi Cs C. [\Psi \triangleright P \mapsto_{iM} \langle N \rangle \prec P'; \bigwedge C. Prop C \Psi P M N P'; (\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P] \implies$
 $Prop C \Psi (Cases Cs) M N P'$

and $rPar1: \bigwedge \Psi \Psi_Q P M N P' A_Q Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM} \langle N \rangle \prec P'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct $A_Q;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P M N P'; distinct A_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* N;$
 $A_Q \#* P'; A_Q \#* C] \implies$
 $Prop C \Psi (P \parallel Q) M N (P' \parallel Q)$

and $rPar2: \bigwedge \Psi \Psi_P Q M N Q' A_P P C.$
 $[\Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} \langle N \rangle \prec Q'; extractFrame P = \langle A_P, \Psi_P \rangle;$
distinct $A_P;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q M N Q'; distinct A_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* N;$
 $A_P \#* Q'; A_P \#* C] \implies$
 $Prop C \Psi (P \parallel Q) M N (P \parallel Q')$

and $rBrMerge: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM} \langle N \rangle \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P M N$
 $P';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} \langle N \rangle \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q M N$
 $Q';$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M] \implies$
 $Prop C \Psi (P \parallel Q) M N (P' \parallel Q')$

and $rScope: \bigwedge \Psi P M N P' x C.$
 $[\Psi \triangleright P \mapsto_{iM} \langle N \rangle \prec P'; \bigwedge C. Prop C \Psi P M N P'; x \# \Psi; x \#$
 $M; x \# N; x \# C] \implies$
 $Prop C \Psi ((\nu x)P) M N ((\nu x)P')$

and $rBang: \bigwedge \Psi P M N P' C.$
 $[\Psi \triangleright P \parallel !P \mapsto_{iM} \langle N \rangle \prec P'; guarded P; \bigwedge C. Prop C \Psi (P \parallel$
 $!P) M N P'] \implies Prop C \Psi (!P) M N P'$

shows $Prop C \Psi P M N P'$
using *Trans*
proof(*nominal-induct* $\Psi P Rs ==_{iM} \langle N \rangle \prec P'$ *avoiding: C arbitrary: P' rule:*
semantics.strong-induct)
case(*cInput* $\Psi M K xvec N Tvec P C$)
then show *?case by (simp add: residualInject)*
next
case(*cBrInput* $\Psi K M xvec N Tvec P C$)
then show *?case*
by(*auto intro: rBrInput simp add: residualInject action.inject*)
next
case(*Output* $\Psi M K N P C$)

```

    then show ?case by(simp add: residualInject)
next
  case BrOutput
  then show ?case by(simp add: residualInject)
next
  case(Case  $\Psi P \varphi CS C P'$ )
  then show ?case by(force intro: rCase)
next
  case(cPar1  $\Psi \Psi_Q P \alpha P' Q A_Q C P''$ )
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case(cPar2  $\Psi \Psi_P Q \alpha Q' xvec P C Q''$ )
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case(cComm1  $\Psi \Psi_Q P M N P' xvec \Psi_P Q K yvec Q' yvec C PQ$ )
  then show ?case by(simp add: residualInject)
next
  case(cComm2  $\Psi \Psi_Q P M zvec N P' xvec \Psi_P Q K yvec Q' C PQ$ )
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(auto intro: rBrMerge simp add: residualInject action.inject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case(cBrClose  $\Psi P M xvec N P' C P''$ )
  then show ?case by(simp add: residualInject)
next
  case(cOpen  $\Psi P M xvec N P' x yvec C P''$ )
  then show ?case by(simp add: residualInject)
next
  case(cBrOpen  $\Psi P M xvec N P' x yvec C P''$ )
  then show ?case by(simp add: residualInject)
next
  case(cScope  $\Psi P \alpha P' x C P''$ )
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case(Bang  $\Psi P C P'$ )
  then show ?case by(force intro: rBang)
qed

```

lemma tauInduct[consumes 1, case-names cCase cPar1 cPar2 cComm1 cComm2 cBrClose cScope cBang]:

```

  fixes  $\Psi$     :: 'b
  and  $P$       :: ('a, 'b, 'c) psi

```

and $R_s :: ('a, 'b, 'c) \text{ residual}$
and $Prop :: 'f::fs\text{-name} \Rightarrow 'b \Rightarrow ('a, 'b, 'c) \text{ psi} \Rightarrow$
 $('a, 'b, 'c) \text{ psi} \Rightarrow \text{bool}$
and $C :: 'f::fs\text{-name}$

assumes $Trans: \Psi \triangleright P \mapsto_{\tau} \prec P'$
and $rCase: \bigwedge \Psi P P' \varphi Cs C. [\Psi \triangleright P \mapsto_{\tau} \prec P'; \bigwedge C. Prop C \Psi P P'; (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P] \Longrightarrow$
 $Prop C \Psi (Cases Cs) P'$
and $rPar1: \bigwedge \Psi \Psi_Q P P' A_Q Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_{\tau} \prec P'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct}$
 $A_Q;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P P';$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi;$
 $A_Q \#* P'; A_Q \#* C] \Longrightarrow$
 $Prop C \Psi (P \parallel Q) (P' \parallel Q)$
and $rPar2: \bigwedge \Psi \Psi_P Q Q' A_P P C.$
 $[\Psi \otimes \Psi_P \triangleright Q \mapsto_{\tau} \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct}$
 $A_P;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q Q';$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi;$
 $A_P \#* Q'; A_P \#* C] \Longrightarrow$
 $Prop C \Psi (P \parallel Q) (P \parallel Q')$
and $rComm1: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q K \text{vec } Q' A_Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
 $\text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * \text{vec}) \langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle; \text{distinct } A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* \text{vec}; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* \text{vec}; \text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* \Psi_Q; \text{vec} \#* P; \text{vec} \#*$
 $M;$
 $\text{vec} \#* Q; \text{vec} \#* K; A_P \#* C; A_Q \#* C; \text{vec} \#* C] \Longrightarrow$
 $Prop C \Psi (P \parallel Q) ((\nu * \text{vec})(P' \parallel Q'))$
and $rComm2: \bigwedge \Psi \Psi_Q P M \text{vec } N P' A_P \Psi_P Q K Q' A_Q C.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * \text{vec}) \langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
 $\text{distinct } A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* \text{vec}; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* \text{vec}; \text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* \Psi_Q; \text{vec} \#* P; \text{vec} \#*$
 $M;$
 $\text{vec} \#* Q; \text{vec} \#* K; A_P \#* C; A_Q \#* C; \text{vec} \#* C] \Longrightarrow$
 $Prop C \Psi (P \parallel Q) ((\nu * \text{vec})(P' \parallel Q'))$

```

and rBrClose:  $\bigwedge \Psi P M \text{vec } N P' A_P \Psi_P x C.$ 
   $\llbracket \Psi \triangleright P \mapsto \text{i}M(\nu * \text{vec}) \langle N \rangle \prec P';$ 
   $x \in \text{supp } M;$ 
   $\text{distinct } \text{vec}; \text{vec} \# * \Psi; \text{vec} \# * P;$ 
   $\text{vec} \# * M;$ 
   $x \# \Psi; x \# \text{vec} \rrbracket \implies$ 
   $\text{Prop } C \Psi ((\nu x)P) ((\nu x)((\nu * \text{vec})P')$ 
and rScope:  $\bigwedge \Psi P P' x C.$ 
   $\llbracket \Psi \triangleright P \mapsto \tau \prec P'; \bigwedge C. \text{Prop } C \Psi P P'; x \# \Psi; x \# C \rrbracket \implies$ 
   $\text{Prop } C \Psi ((\nu x)P) ((\nu x)P')$ 
and rBang:  $\bigwedge \Psi P P' C.$ 
   $\llbracket \Psi \triangleright P \parallel !P \mapsto \tau \prec P'; \text{guarded } P; \bigwedge C. \text{Prop } C \Psi (P \parallel !P) P \rrbracket$ 
 $\implies \text{Prop } C \Psi (!P) P'$ 
shows  $\text{Prop } C \Psi P P'$ 
  using Trans
proof(nominal-induct  $\Psi P R s == \tau \prec P'$  avoiding: C arbitrary: P' rule: semantics.strong-induct)
  case(cInput  $M K \text{vec } N T \text{vec } P C$ )
  then show ?case by(simp add: residualInject)
next
  case cBrInput
  then show ?case by(simp add: residualInject)
next
  case(Output  $\Psi M K N P C$ )
  then show ?case by(simp add: residualInject)
next
  case BrOutput
  then show ?case by(simp add: residualInject)
next
  case(Case  $\Psi P \varphi C s C P'$ )
  then show ?case by(force intro: rCase simp add: residualInject)
next
  case(cPar1  $\Psi \Psi_Q P \alpha P' A_Q Q C P''$ )
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case(cPar2  $\Psi \Psi_P Q \alpha Q' A_P P C Q''$ )
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case(cComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q K \text{vec } Q' A_Q C P Q$ )
  then show ?case by(force intro: rComm1 simp add: residualInject)
next
  case(cComm2  $\Psi \Psi_Q P M \text{vec } N P' A_P \Psi_P Q' A_Q C P Q$ )
  then show ?case by(force intro: rComm2 simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)

```

```

next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(force intro: rBrClose simp add: residualInject)
next
  case(cOpen Ψ P M xvec N P' x yvec C P'')
  then show ?case by(simp add: residualInject)
next
  case(cBrOpen Ψ P M xvec N P' x yvec C P'')
  then show ?case by(simp add: residualInject)
next
  case(cScope Ψ P α P' x C P'')
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case(Bang Ψ P C P')
  then show ?case by(force intro: rBang simp add: residualInject)
qed

```

lemma *semanticsFrameInduct*[consumes 3, case-names *cAlpha cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:

```

fixes Ψ    :: 'b
and P      :: ('a, 'b, 'c) psi
and Rs     :: ('a, 'b, 'c) residual
and AP    :: name list
and ΨP    :: 'b
and Prop   :: 'f::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒
              ('a, 'b, 'c) residual ⇒ name list ⇒ 'b ⇒ bool
and C      :: 'f::fs-name

```

assumes *Trans*: $\Psi \triangleright P \mapsto Rs$

```

and FrP: extractFrame P = ⟨AP, ΨP⟩
and distinct AP
and rAlpha:  $\bigwedge \Psi P A_P \Psi_P p Rs C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* (p \cdot A_P); A_P \#* Rs; A_P \#* C \rrbracket$ ;

```

$$\text{set } p \subseteq \text{set } A_P \times \text{set}(p \cdot A_P); \text{distinctPerm } p;$$

$$\text{Prop } C \Psi P Rs A_P \Psi_P \rrbracket \implies \text{Prop } C \Psi P Rs (p$$

$\cdot A_P) (p \cdot \Psi_P)$

```

and rInput:  $\bigwedge \Psi M K xvec N Tvec P C.$ 
 $\llbracket \Psi \vdash M \leftrightarrow K; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N;$ 
 $\text{length } xvec = \text{length } Tvec; xvec \#* \Psi;$ 
 $xvec \#* M; xvec \#* K; xvec \#* C \rrbracket \implies$ 
 $\text{Prop } C \Psi (M(\lambda * xvec N).P)$ 
 $(K(\llbracket N[xvec::=Tvec] \rrbracket)) \prec (P[xvec::=Tvec]) \rrbracket \quad (\square) \quad (1)$ 

```

```

and rBrInput:  $\bigwedge \Psi M K xvec N Tvec P C.$ 
 $\llbracket \Psi \vdash K \succeq M; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N;$ 
 $\text{length } xvec = \text{length } Tvec; xvec \#* \Psi;$ 

```

$$\begin{aligned}
& xvec \#* M; xvec \#* K; xvec \#* C \Longrightarrow \\
& Prop C \Psi (M(\lambda*xvec N).P) \\
& (\downarrow K(\langle N[xvec::=Tvec] \rangle) \prec (P[xvec::=Tvec])) (\square) \mathbf{(1)} \\
\text{and } rOutput: & \bigwedge \Psi M K N P C. \Psi \vdash M \leftrightarrow K \Longrightarrow Prop C \Psi (M\langle N \rangle.P) (K\langle N \rangle \\
& \prec P) (\square) \mathbf{(1)} \\
\text{and } rBrOutput: & \bigwedge \Psi M K N P C. \Psi \vdash M \preceq K \Longrightarrow Prop C \Psi (M\langle N \rangle.P) \\
& (\downarrow K\langle N \rangle \prec P) (\square) \mathbf{(1)} \\
\text{and } rCase: & \bigwedge \Psi P Rs \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto Rs; extractFrame P = \langle A_P, \\
& \Psi_P \rangle; distinct A_P; \bigwedge C. Prop C \Psi P Rs A_P \Psi_P; \\
& (\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P; \Psi_P \simeq \mathbf{1}; \\
& (supp \Psi_P) = (\{\}::name set); \\
& A_P \#* \Psi; A_P \#* P; A_P \#* Rs; A_P \#* C \Longrightarrow \\
Prop C \Psi (Cases Cs) Rs (\square) \mathbf{(1)} \\
\text{and } rPar1: & \bigwedge \Psi \Psi_Q P \alpha P' A_Q Q A_P \Psi_P C. \\
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'; \\
& extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \\
& extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q; \\
& \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P (\alpha \prec P') A_P \Psi_P; distinct(bn \alpha); \\
& A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* P'; A_P \#* A_Q; A_P \\
& \#* \Psi_Q; \\
& A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* P'; A_Q \#* \Psi_P; \\
& bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* subject \alpha; bn \alpha \#* \Psi_P; \\
bn \alpha \#* \Psi_Q; \\
& A_P \#* C; A_Q \#* C; bn \alpha \#* C \Longrightarrow \\
Prop C \Psi (P \parallel Q) (\alpha \prec (P' \parallel Q)) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q) \\
\text{and } rPar2: & \bigwedge \Psi \Psi_P Q \alpha Q' A_P P A_Q \Psi_Q C. \\
& \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'; \\
& extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \\
& extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q; \\
& \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (\alpha \prec Q') A_Q \Psi_Q; distinct(bn \alpha); \\
& A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* Q'; A_P \#* A_Q; A_P \\
& \#* \Psi_Q; \\
& A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* Q'; A_Q \#* \Psi_P; \\
& bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* subject \alpha; bn \alpha \#* \Psi_P; \\
bn \alpha \#* \Psi_Q; \\
& A_P \#* C; A_Q \#* C; bn \alpha \#* C \Longrightarrow \\
Prop C \Psi (P \parallel Q) (\alpha \prec (P \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q) \\
\text{and } rComm1: & \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q C. \\
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle N \rangle) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; \\
distinct A_P; \\
& \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P ((M(\langle N \rangle)) \prec P') A_P \Psi_P; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto K(\downarrow \nu*xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \\
\Psi_Q \rangle; distinct A_Q; \\
& \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; \\
\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (K(\downarrow \nu*xvec)\langle N \rangle \prec Q') A_Q \Psi_Q; distinct \\
xvec; \\
& A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; \\
& A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P; \\
& A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';
\end{aligned}$$

$A_Q \#* \text{vec}; \text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* \Psi_Q; \text{vec} \#* P; \text{vec} \#*$
 $M;$
 $\text{vec} \#* Q; \text{vec} \#* K; A_P \#* C; A_Q \#* C; \text{vec} \#* C] \implies$
 $\text{Prop } C \Psi (P \parallel Q) (\tau \prec (\nu * \text{vec})(P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rComm2: \bigwedge \Psi \Psi_Q P M \text{vec} N P' A_P \Psi_P Q K Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * \text{vec})(N) \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P (M(\nu * \text{vec})(N) \prec P') A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
 $\text{distinct } A_Q;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q (K(N) \prec Q') A_Q \Psi_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; \text{distinct } \text{vec};$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* \text{vec}; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* \text{vec}; \text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* \Psi_Q; \text{vec} \#* P; \text{vec} \#*$
 $M;$
 $\text{vec} \#* Q; \text{vec} \#* K; A_P \#* C; A_Q \#* C; \text{vec} \#* C] \implies$
 $\text{Prop } C \Psi (P \parallel Q) (\tau \prec (\nu * \text{vec})(P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rBrMerge: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(N) \prec P'; \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P$
 $(iM(N) \prec P') A_P \Psi_P;$
 $\text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q'; \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q$
 $(iM(N) \prec Q') A_Q \Psi_Q;$
 $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M] \implies$
 $\text{Prop } C \Psi (P \parallel Q) (iM(N) \prec (P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rBrComm1: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q \text{vec} Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(N) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
 $\text{distinct } A_P;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P ((iM(N)) \prec P') A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(\nu * \text{vec})(N) \prec Q'; \text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle; \text{distinct } A_Q;$
 $\bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q (iM(\nu * \text{vec})(N) \prec Q') A_Q \Psi_Q; \text{distinct}$
 $\text{vec};$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* \text{vec}; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q';$
 $A_Q \#* \text{vec}; \text{vec} \#* \Psi; \text{vec} \#* \Psi_P; \text{vec} \#* \Psi_Q; \text{vec} \#* P;$
 $\text{vec} \#* Q; A_P \#* C; A_Q \#* C; \text{vec} \#* C;$
 $A_P \#* M; A_Q \#* M; \text{vec} \#* M] \implies$
 $\text{Prop } C \Psi (P \parallel Q) (iM(\nu * \text{vec})(N) \prec (P' \parallel Q')) (A_P @ A_Q) (\Psi_P$
 $\otimes \Psi_Q)$
and $rBrComm2: \bigwedge \Psi \Psi_Q P M \text{vec} N P' A_P \Psi_P Q Q' A_Q C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \text{iM}(\nu^* \text{xvec}) \langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\wedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P (\text{iM}(\nu^* \text{xvec}) \langle N \rangle \prec P') A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \text{iM} \langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
 $\text{distinct } A_Q;$
 $\wedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q (\text{iM} \langle N \rangle \prec Q') A_Q \Psi_Q; \text{distinct } \text{xvec};$
 $A_P \#^* \Psi; A_P \#^* \Psi_Q; A_P \#^* P; A_P \#^* N; A_P \#^* P';$
 $A_P \#^* Q; A_P \#^* Q'; A_P \#^* A_Q; A_P \#^* \text{xvec}; A_Q \#^* \Psi; A_Q \#^* \Psi_P;$
 $A_Q \#^* P; A_Q \#^* N; A_Q \#^* P'; A_Q \#^* Q; A_Q \#^* Q';$
 $A_Q \#^* \text{xvec}; \text{xvec} \#^* \Psi; \text{xvec} \#^* \Psi_P; \text{xvec} \#^* \Psi_Q; \text{xvec} \#^* P;$
 $\text{xvec} \#^* Q; A_P \#^* C; A_Q \#^* C; \text{xvec} \#^* C;$
 $A_P \#^* M; A_Q \#^* M; \text{xvec} \#^* M \rrbracket \implies$
 $\text{Prop } C \Psi (P \parallel Q) (\text{iM}(\nu^* \text{xvec}) \langle N \rangle \prec (P' \parallel Q')) (A_P @ A_Q) (\Psi_P$
 $\otimes \Psi_Q)$
and $rBrClose: \wedge \Psi P M \text{xvec } N P' A_P \Psi_P x C.$
 $\llbracket \Psi \triangleright P \mapsto \text{iM}(\nu^* \text{xvec}) \langle N \rangle \prec P';$
 $x \in \text{supp } M;$
 $\wedge C. \text{Prop } C \Psi P (\text{iM}(\nu^* \text{xvec}) \langle N \rangle \prec P') A_P \Psi_P;$
 $\text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* N; A_P \#^* P'; A_P \#^* \text{xvec};$
 $\text{distinct } \text{xvec}; \text{xvec} \#^* \Psi; \text{xvec} \#^* \Psi_P; \text{xvec} \#^* P;$
 $\text{xvec} \#^* M;$
 $x \# \Psi; x \# \text{xvec}; x \# A_P;$
 $A_P \#^* C; \text{xvec} \#^* C; x \# C \rrbracket \implies$
 $\text{Prop } C \Psi ((\nu x)P) (\tau \prec ((\nu x)((\nu^* \text{xvec})P')) (x \# A_P) \Psi_P$
and $rOpen: \wedge \Psi P M \text{xvec } yvec N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto M(\nu^*(\text{xvec} @ yvec)) \langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\wedge C. \text{Prop } C \Psi P (M(\nu^*(\text{xvec} @ yvec)) \langle N \rangle \prec P') A_P \Psi_P; x \in \text{supp}$
 $N; x \# \Psi; x \# M;$
 $x \# A_P; x \# \text{xvec}; x \# yvec; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^*$
 $N; A_P \#^* P';$
 $A_P \#^* \text{xvec}; A_P \#^* yvec; \text{xvec} \#^* yvec; \text{distinct } \text{xvec}; \text{distinct } yvec;$
 $\text{xvec} \#^* \Psi; \text{xvec} \#^* P; \text{xvec} \#^* M; \text{xvec} \#^* \Psi_P; yvec \#^* \Psi_P;$
 $yvec \#^* \Psi; yvec \#^* P; yvec \#^* M; A_P \#^* C; x \# C; \text{xvec} \#^* C; yvec$
 $\#^* C \rrbracket \implies$
 $\text{Prop } C \Psi ((\nu x)P) (M(\nu^*(\text{xvec} @ x \# yvec)) \langle N \rangle \prec P') (x \# A_P) \Psi_P$
and $rBrOpen: \wedge \Psi P M \text{xvec } yvec N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto \text{iM}(\nu^*(\text{xvec} @ yvec)) \langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\wedge C. \text{Prop } C \Psi P (\text{iM}(\nu^*(\text{xvec} @ yvec)) \langle N \rangle \prec P') A_P \Psi_P; x \in$
 $\text{supp } N; x \# \Psi; x \# M;$
 $x \# A_P; x \# \text{xvec}; x \# yvec; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^*$
 $N; A_P \#^* P';$
 $A_P \#^* \text{xvec}; A_P \#^* yvec; \text{xvec} \#^* yvec; \text{distinct } \text{xvec}; \text{distinct } yvec;$
 $\text{xvec} \#^* \Psi; \text{xvec} \#^* P; \text{xvec} \#^* M; \text{xvec} \#^* \Psi_P; yvec \#^* \Psi_P;$
 $yvec \#^* \Psi; yvec \#^* P; yvec \#^* M; A_P \#^* C; x \# C; \text{xvec} \#^* C; yvec$
 $\#^* C \rrbracket \implies$
 $\text{Prop } C \Psi ((\nu x)P) (\text{iM}(\nu^*(\text{xvec} @ x \# yvec)) \langle N \rangle \prec P') (x \# A_P) \Psi_P$

```

and rScope:  $\bigwedge \Psi P \alpha P' x A_P \Psi_P C.$ 
   $\llbracket \Psi \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$ 
   $\bigwedge C. \text{Prop } C \Psi P (\alpha \prec P') A_P \Psi_P;$ 
   $x \# \Psi; x \# \alpha; x \# A_P; A_P \#* \Psi; A_P \#* P;$ 
   $A_P \#* \alpha; A_P \#* P'; \text{distinct}(bn \alpha);$ 
   $bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* \text{subject } \alpha; bn \alpha \#* \Psi_P;$ 
   $A_P \#* C; x \# C; bn \alpha \#* C \rrbracket \implies$ 
   $\text{Prop } C \Psi ((\nu x)P) (\alpha \prec ((\nu x)P')) (x \# A_P) \Psi_P$ 
and rBang:  $\bigwedge \Psi P Rs A_P \Psi_P C.$ 
   $\llbracket \Psi \triangleright P \parallel !P \mapsto Rs; \text{guarded } P; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$ 
distinct  $A_P;$ 
   $\bigwedge C. \text{Prop } C \Psi (P \parallel !P) Rs A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; \text{supp } \Psi_P =$ 
   $(\{\}::\text{name set});$ 
   $A_P \#* \Psi; A_P \#* P; A_P \#* Rs; A_P \#* C \rrbracket \implies \text{Prop } C \Psi (!P) Rs$ 
 $(\{\})$  (1)
shows  $\text{Prop } C \Psi P Rs A_P \Psi_P$ 
  using Trans FrP  $\langle \text{distinct } A_P \rangle$ 
proof(nominal-induct avoiding:  $A_P \Psi_P C$  rule: semantics.strong-induct)
  case(cInput  $\Psi M K xvec N Tvec P A_P \Psi_P C$ )
  from  $\langle \text{extractFrame } (M(\lambda * xvec N)).P \rangle = \langle A_P, \Psi_P \rangle$ 
  have  $A_P = \{\}$  and  $\Psi_P = \mathbf{1}$ 
  by auto
  with  $\langle \Psi \vdash M \leftrightarrow K \rangle \langle \text{distinct } xvec \rangle \langle \text{set } xvec \subseteq \text{supp } N \rangle \langle \text{length } xvec = \text{length}$ 
Tvec
   $\langle xvec \#* \Psi \rangle \langle xvec \#* M \rangle \langle xvec \#* K \rangle \langle xvec \#* C \rangle$ 
  show ?case by(blast intro: rInput)
next
  case(cBrInput  $\Psi K M xvec N Tvec P A_P \Psi_P C$ )
  from  $\langle \text{extractFrame } (M(\lambda * xvec N)).P \rangle = \langle A_P, \Psi_P \rangle$ 
  have  $A_P = \{\}$  and  $\Psi_P = \mathbf{1}$ 
  by auto
  with  $\langle \Psi \vdash K \succeq M \rangle \langle \text{distinct } xvec \rangle \langle \text{set } xvec \subseteq \text{supp } N \rangle \langle \text{length } xvec = \text{length}$ 
Tvec
   $\langle xvec \#* \Psi \rangle \langle xvec \#* M \rangle \langle xvec \#* K \rangle \langle xvec \#* C \rangle$ 
  show ?case by(blast intro: rBrInput)
next
  case(Output  $\Psi M K N P A_P \Psi_P$ )
  from  $\langle \text{extractFrame } (M(N)).P \rangle = \langle A_P, \Psi_P \rangle$ 
  have  $A_P = \{\}$  and  $\Psi_P = \mathbf{1}$ 
  by auto
  with  $\langle \Psi \vdash M \leftrightarrow K \rangle$  show ?case
  by(blast intro: rOutput)
next
  case(BrOutput  $\Psi M K N P A_P \Psi_P$ )
  from  $\langle \text{extractFrame } (M(N)).P \rangle = \langle A_P, \Psi_P \rangle$ 
  have  $A_P = \{\}$  and  $\Psi_P = \mathbf{1}$ 
  by auto
  with  $\langle \Psi \vdash M \preceq K \rangle$  show ?case
  by(blast intro: rBrOutput)

```

next
case(*Case* $\Psi P Rs \varphi Cs A_{cP} \Psi_{cP} C$)
obtain $A_P \Psi_P$ **where** *FrP*: $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
and $A_P \#^* (\Psi, P, Rs, C)$
by(*rule freshFrame*)
then have $A_P \#^* \Psi$ **and** $A_P \#^* P$ **and** $A_P \#^* Rs$ **and** $A_P \#^* C$
by *simp*+
note $\langle \Psi \triangleright P \mapsto Rs \rangle \text{FrP} \langle \text{distinct } A_P \rangle$
moreover from *FrP* $\langle \text{distinct } A_P \rangle \langle \bigwedge A_P \Psi_P C. \llbracket \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P \rrbracket \implies \text{Prop } C \Psi P Rs A_P \Psi_P \rangle$
have $\bigwedge C. \text{Prop } C \Psi P Rs A_P \Psi_P$ **by** *simp*
moreover note $\langle (\varphi, P) \in \text{set } Cs \rangle \langle \Psi \vdash \varphi \rangle \langle \text{guarded } P \rangle$
moreover from $\langle \text{guarded } P \rangle \text{FrP}$ **have** $\Psi_P \simeq \mathbf{1}$ **and** $\text{supp } \Psi_P = (\{\} :: \text{name set})$
by(*metis guardedStatEq*)+
moreover note $\langle A_P \#^* \Psi \rangle \langle A_P \#^* P \rangle \langle A_P \#^* Rs \rangle \langle A_P \#^* C \rangle$
ultimately have $\text{Prop } C \Psi (Cases Cs) Rs (\square) (\mathbf{1})$
by(*rule rCase*)
then show *?case* **using** $\langle \text{extractFrame}(Cases Cs) = \langle A_{cP}, \Psi_{cP} \rangle \rangle$ **by** *simp*
next
case(*cPar1* $\Psi \Psi_Q P \alpha P' Q A_Q A_{PQ} \Psi_{PQ} C$)
obtain $A_P \Psi_P$ **where** *FrP*: $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
 $A_P \#^* (P, Q, \Psi, \alpha, P', A_Q, A_{PQ}, C, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \#^* P$ **and** $A_P \#^* Q$ **and** $A_P \#^* \Psi$ **and** $A_P \#^* \alpha$ **and** $A_P \#^* P'$
and $A_P \#^* A_Q$ **and** $A_P \#^* A_{PQ}$ **and** $A_P \#^* C$ **and** $A_P \#^* \Psi_Q$
by *simp*+

have *FrQ*: $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_Q \#^* P \rangle \langle A_P \#^* A_Q \rangle \text{FrP}$ **have** $A_Q \#^* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle \text{bn } \alpha \#^* P \rangle \langle A_P \#^* \alpha \rangle \text{FrP}$ **have** $\text{bn } \alpha \#^* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \text{FrP} \text{FrQ} \langle A_P \#^* A_Q \rangle \langle A_P \#^* \Psi_Q \rangle$
 $\langle A_Q \#^* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#^* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and *distinctPerm* p
and $\Psi_{PQ}: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $A_{PQ}: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#^* A_{PQ} \rangle \langle A_Q \#^* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
apply –
apply(*rule frameChainEq'*)
by (*assumption* | *simp add: eqvts*)+
note $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle \text{FrP} \langle \text{distinct } A_P \rangle \text{FrQ} \langle \text{distinct } A_Q \rangle$

moreover from $FrP \langle distinct A_P \rangle \langle \wedge A_P \Psi_P C. \llbracket extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P \rrbracket \implies Prop C (\Psi \otimes \Psi_Q) P (\alpha \prec P') A_P \Psi_P \rangle$
have $\wedge C. Prop C (\Psi \otimes \Psi_Q) P (\alpha \prec P') A_P \Psi_P$ **by** *simp*

moreover note $\langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \alpha \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle distinct(bn \alpha) \rangle$
 $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* \Psi_P \rangle \langle bn \alpha \#* \Psi_Q \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle bn \alpha \#* C \rangle$
ultimately have $Prop C \Psi (P \parallel Q) (\alpha \prec (P' \parallel Q)) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
by *(metis rPar1)*

with $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle A_P \#* A_{PQ} \rangle \langle A_P \#* C \rangle$
 $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \alpha \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* A_{PQ} \rangle \langle A_Q \#* C \rangle$
 $S \langle distinctPerm p \rangle Aeq$
have $Prop C \Psi (P \parallel Q) (\alpha \prec (P' \parallel Q)) (p \cdot (A_P @ A_Q)) (p \cdot (\Psi_P \otimes \Psi_Q))$
apply $-$
apply *(rule rAlpha)*
by *(assumption | simp add: eqvts)+*
with $\Psi eq Aeq$ **show** *?case by (simp add: eqvts)*

next
case *(cPar2 $\Psi \Psi_P Q \alpha Q' P A_P A_{PQ} \Psi_{PQ} C$)*
obtain $A_Q \Psi_Q$ **where** $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** *distinct A_Q*
 $A_Q \#* (P, Q, \Psi, \alpha, Q', A_P, A_{PQ}, C, \Psi_P)$
by *(rule freshFrame)*
then have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* \alpha$ **and** $A_Q \#* Q'$
and $A_Q \#* A_P$ **and** $A_Q \#* A_{PQ}$ **and** $A_Q \#* C$ **and** $A_Q \#* \Psi_P$
by *simp+*

from $\langle A_Q \#* A_P \rangle$ **have** $A_P \#* A_Q$ **by** *simp*
have $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ FrQ **have** $A_P \#* \Psi_Q$
by *(force dest: extractFrameFreshChain)*
from $\langle bn \alpha \#* Q \rangle \langle A_Q \#* \alpha \rangle$ FrQ **have** $bn \alpha \#* \Psi_Q$
by *(force dest: extractFrameFreshChain)*

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ $FrP FrQ \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** *distinct $(A_P @ A_Q)$*
by *(auto simp add: fresh-star-def fresh-def name-list-supp)*
ultimately obtain p **where** $S: (set p \subseteq (set(A_P @ A_Q)) \times (set A_{PQ}))$ **and** *distinctPerm p*

and $\Psi eq: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $Aeq: APQ = ((p \cdot AP) @ (p \cdot AQ))$
using $\langle AP \#* APQ \rangle \langle AQ \#* APQ \rangle \langle distinct APQ \rangle$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp add: eqvts*)+

note $\langle \Psi \otimes \Psi_P \triangleright Q \longmapsto \alpha \prec Q' \rangle FrP \langle distinct AP \rangle FrQ \langle distinct AQ \rangle$

moreover from $FrQ \langle distinct AQ \rangle \langle \bigwedge AQ \Psi_Q C. \llbracket extractFrame Q = \langle AQ, \Psi_Q \rangle; distinct AQ \rrbracket \implies Prop C (\Psi \otimes \Psi_P) Q (\alpha \prec Q') AQ \Psi_Q$
have $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (\alpha \prec Q') AQ \Psi_Q$ **by** *simp*

moreover note $\langle AP \#* P \rangle \langle AP \#* Q \rangle \langle AP \#* \Psi \rangle \langle AP \#* \alpha \rangle \langle AP \#* Q' \rangle \langle AP \#* APQ \rangle \langle AP \#* \Psi_Q \rangle$
 $\langle AQ \#* P \rangle \langle AQ \#* Q \rangle \langle AQ \#* \Psi \rangle \langle AQ \#* \alpha \rangle \langle AQ \#* Q' \rangle \langle AQ \#* \Psi_P \rangle \langle distinct (bn \alpha) \rangle$
 $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* \Psi_P \rangle \langle bn \alpha \#* \Psi_Q \rangle$
 $\langle AP \#* C \rangle \langle AQ \#* C \rangle \langle bn \alpha \#* C \rangle$
ultimately have $Prop C \Psi (P \parallel Q) (\alpha \prec (P \parallel Q')) (AP @ AQ) (\Psi_P \otimes \Psi_Q)$
by(*metis rPar2*)

with $\langle AP \#* \Psi \rangle \langle AP \#* P \rangle \langle AP \#* Q \rangle \langle AP \#* \alpha \rangle \langle AP \#* Q' \rangle \langle AP \#* APQ \rangle \langle AP \#* C \rangle$
 $\langle AQ \#* \Psi \rangle \langle AQ \#* P \rangle \langle AQ \#* Q \rangle \langle AQ \#* \alpha \rangle \langle AQ \#* Q' \rangle \langle AQ \#* APQ \rangle \langle AQ \#* C \rangle$
 $S \langle distinctPerm p \rangle Aeq$
have $Prop C \Psi (P \parallel Q) (\alpha \prec (P \parallel Q')) (p \cdot (AP @ AQ)) (p \cdot (\Psi_P \otimes \Psi_Q))$
apply –
apply(*rule rAlpha*)
by(*assumption | simp add: eqvts*)+
with $\Psi eq Aeq$ **show** *?case* **by**(*simp add: eqvts*)

next
case(*cComm1* $\Psi \Psi_Q P M N P' AP \Psi_P Q K xvec Q' AQ APQ \Psi_{PQ} C$)
from $\langle distinct AP \rangle \langle distinct AQ \rangle \langle AP \#* AQ \rangle$ **have** $distinct (AP @ AQ)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
from *cComm1* **have** $Prop C \Psi (P \parallel Q) (\tau \prec (\nu * xvec)(P' \parallel Q')) (AP @ AQ)$
 $(\Psi_P \otimes \Psi_Q)$
by(*metis rComm1*)
moreover from $\langle extractFrame (P \parallel Q) = \langle APQ, \Psi_{PQ} \rangle \rangle \langle extractFrame P = \langle AP, \Psi_P \rangle \rangle \langle extractFrame Q = \langle AQ, \Psi_Q \rangle \rangle$
 $\langle AP \#* AQ \rangle \langle AP \#* \Psi_Q \rangle \langle AQ \#* \Psi_P \rangle$
have $\langle (AP @ AQ), (\Psi_P \otimes \Psi_Q) \rangle = \langle APQ, \Psi_{PQ} \rangle$
by *simp*
with $\langle AP \#* APQ \rangle \langle AQ \#* APQ \rangle \langle distinct (AP @ AQ) \rangle \langle distinct APQ \rangle$
obtain p **where** $S: (set p \subseteq (set (AP @ AQ)) \times (set APQ))$ **and** $distinctPerm p$
and $\Psi eq: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $Aeq: APQ = p \cdot (AP @ AQ)$
apply –
apply(*rule frameChainEq'*)

by(*assumption | simp*)
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P \#* xvec \rangle$
 $\langle A_Q \#* xvec \rangle \langle A_P \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* APQ \rangle \langle A_Q \#* APQ \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle$
ultimately have *Prop C* $\Psi (P \parallel Q) (\tau \prec (\nu*xvec)(P' \parallel Q')) (p \cdot (A_P@A_Q))$
 $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce simp add: rAlpha*)
with $\Psi eq Aeq$ **show** ?*case by simp*
next
case(*cComm2* $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q APQ \Psi_{PQ} C$)
from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $distinct(A_P@A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
from *cComm2* **have** *Prop C* $\Psi (P \parallel Q) (\tau \prec (\nu*xvec)(P' \parallel Q')) (A_P@A_Q)$
 $(\Psi_P \otimes \Psi_Q)$
by(*metis rComm2*)
moreover from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P@A_Q), (\Psi_P \otimes \Psi_Q) \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*
with $\langle A_P \#* APQ \rangle \langle A_Q \#* APQ \rangle \langle distinct(A_P@A_Q) \rangle \langle distinct APQ \rangle$
obtain *p* **where** *S*: $(set p \subseteq (set(A_P@A_Q)) \times (set APQ))$ **and** *distinctPerm p*
and $\Psi eq: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $Aeq: APQ = p \cdot (A_P@A_Q)$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*)
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P \#* xvec \rangle$
 $\langle A_Q \#* xvec \rangle \langle A_P \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* APQ \rangle \langle A_Q \#* APQ \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle$
ultimately have *Prop C* $\Psi (P \parallel Q) (\tau \prec (\nu*xvec)(P' \parallel Q')) (p \cdot (A_P@A_Q))$
 $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro: rAlpha*)
with $\Psi eq Aeq$ **show** ?*case by simp*
next
case(*cBrMerge* $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q APQ \Psi_{PQ} C$)
from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $distinct(A_P@A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
from *cBrMerge* **have** *Prop C* $\Psi (P \parallel Q) (iM(N) \prec (P' \parallel Q')) (A_P@A_Q) (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro!: rBrMerge*)
moreover from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P@A_Q), (\Psi_P \otimes \Psi_Q) \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*

with $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct}(A_P @ A_Q) \rangle \langle \text{distinct } A_{PQ} \rangle$
obtain p **where** $S: (\text{set } p \subseteq (\text{set}(A_P @ A_Q)) \times (\text{set } A_{PQ}))$ **and** $\text{distinctPerm } p$
and $\Psi eq: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $Aeq: A_{PQ} = p \cdot (A_P @ A_Q)$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*)**+**
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#*$
 $Q \rangle$
 $\langle A_P \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* N \rangle \langle A_Q \#* N \rangle$
ultimately have $\text{Prop } C \Psi (P \parallel Q) (\text{iM}(\nu N) \prec (P' \parallel Q')) (p \cdot (A_P @ A_Q)) (p$
 $\cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro: rAlpha*)
with Ψeq Aeq **show** *?case by simp*
next
case(*cBrComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q \text{xvec } Q' A_Q A_{PQ} \Psi_{PQ} C$)
from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
from *cBrComm1* **have** $\text{Prop } C \Psi (P \parallel Q) (\text{iM}(\nu \text{xvec}) \langle N \rangle \prec (P' \parallel Q'))$
 $(A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
by(*metis rBrComm1*)
moreover from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \langle \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), (\Psi_P \otimes \Psi_Q) \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*
with $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct}(A_P @ A_Q) \rangle \langle \text{distinct } A_{PQ} \rangle$
obtain p **where** $S: (\text{set } p \subseteq (\text{set}(A_P @ A_Q)) \times (\text{set } A_{PQ}))$ **and** $\text{distinctPerm } p$
and $\Psi eq: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $Aeq: A_{PQ} = p \cdot (A_P @ A_Q)$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*)**+**
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#*$
 $Q \rangle \langle A_P \#* \text{xvec} \rangle$
 $\langle A_Q \#* \text{xvec} \rangle \langle A_P \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* A_{PQ} \rangle \langle A_Q$
 $\#* A_{PQ} \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* N \rangle \langle A_Q \#* N \rangle$
ultimately have $\text{Prop } C \Psi (P \parallel Q) (\text{iM}(\nu \text{xvec}) \langle N \rangle \prec (P' \parallel Q')) (p \cdot (A_P @ A_Q))$
 $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro: rAlpha*)
with Ψeq Aeq **show** *?case by simp*
next
case(*cBrComm2* $\Psi \Psi_Q P M \text{xvec } N P' A_P \Psi_P Q Q' A_Q A_{PQ} \Psi_{PQ} C$)
from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
from *cBrComm2* **have** $\text{Prop } C \Psi (P \parallel Q) (\text{iM}(\nu \text{xvec}) \langle N \rangle \prec (P' \parallel Q'))$
 $(A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
by(*metis rBrComm2*)
moreover from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \langle \text{extractFrame } P = \langle A_P,$

$\Psi_P\rangle \langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), (\Psi_P \otimes \Psi_Q) \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$
by *simp*
with $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle distinct(A_P @ A_Q) \rangle \langle distinct\ A_{PQ} \rangle$
obtain p **where** $S: (set\ p \subseteq (set(A_P @ A_Q)) \times (set\ A_{PQ}))$ **and** $distinctPerm\ p$
and $\Psi_{eq}: \Psi_{PQ} = p \cdot (\Psi_P \otimes \Psi_Q)$ **and** $A_{eq}: A_{PQ} = p \cdot (A_P @ A_Q)$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*)**+**
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#*$
 $Q \rangle \langle A_P \#* xvec \rangle$
 $\langle A_Q \#* xvec \rangle \langle A_P \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* A_{PQ} \rangle \langle A_Q$
 $\#* A_{PQ} \rangle$
 $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* N \rangle \langle A_Q \#* N \rangle$
ultimately have $Prop\ C\ \Psi\ (P \parallel Q)\ (\downarrow M(\downarrow \nu * xvec)\langle N \rangle \prec (P' \parallel Q'))\ (p \cdot (A_P @ A_Q))$
 $(p \cdot (\Psi_P \otimes \Psi_Q))$
by(*fastforce intro: rAlpha*)
with $\Psi_{eq}\ A_{eq}$ **show** *?case by simp*
next
case(*cBrClose* $\Psi\ P\ M\ xvec\ N\ P'\ x\ A_{P'}\ \Psi_{P'}\ C$)
obtain $A_P\ \Psi_P$ **where** $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **and** $distinct\ A_P$
and $A_P \#* (\Psi, P, M, xvec, N, P', A_{P'}, \Psi_{P'}, C, x)$
by(*rule freshFrame*)
then have $A_P \#* \Psi$ **and** $A_P \#* P$ **and** $A_P \#* M$ **and** $A_P \#* xvec$ **and** $A_P \#* N$
and $A_P \#* P'$
and $A_P \#* A_{P'}$ **and** $A_P \#* \Psi_{P'}$ **and** $A_P \#* C$ **and** $x \# A_P$
by *simp***+**
from $FrP\ \langle A_P \#* xvec \rangle \langle xvec \#* P \rangle$ **have** $xvec \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)
from $\langle A_P \#* xvec \rangle \langle A_P \#* P' \rangle \langle x \# A_P \rangle$
have $A_P \#* (\tau \prec (\downarrow \nu x)((\downarrow \nu * xvec)P'))$ **by** *simp*
from $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle \langle extractFrame\ ((\downarrow \nu x)P) = \langle A_{P'}, \Psi_{P'} \rangle \rangle$
have $\langle (x \# A_P), \Psi_P \rangle = \langle A_{P'}, \Psi_{P'} \rangle$ **by** *simp*
with $\langle A_P \#* A_{P'} \rangle \langle x \# A_{P'} \rangle \langle x \# A_P \rangle \langle distinct\ A_P \rangle \langle distinct\ A_{P'} \rangle$
obtain p **where** $S: (set\ p \subseteq (set\ (x \# A_P)) \times (set\ A_{P'}))$ **and** $distinctPerm\ p$
and $\Psi_{eq}: \Psi_{P'} = p \cdot \Psi_P$ **and** $A_{eq}: A_{P'} = p \cdot (x \# A_P)$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*)**+**

from $\langle x \# A_{P'} \rangle\ A_{eq}$ **have** $x \# (p \cdot (x \# A_P))$ **by** *simp*
moreover from $S\ A_{eq}\ \langle distinct\ A_{P'} \rangle \langle x \# A_{P'} \rangle \langle A_P \#* A_{P'} \rangle$ **have** $(p \cdot x) \# A_P$
by *simp*
from *cBrClose* $FrP\ \langle distinct\ A_P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle$
 $\langle A_P \#* N \rangle \langle A_P \#* P' \rangle$
 $\langle A_P \#* A_{P'} \rangle \langle A_P \#* \Psi_{P'} \rangle \langle A_P \#* C \rangle \langle x \# A_P \rangle \langle xvec \#* \Psi_P \rangle \langle A_P \#* C \rangle \langle xvec$
 $\#* C \rangle \langle x \# C \rangle$
have $Prop\ C\ \Psi\ ((\downarrow \nu x)P)\ (\tau \prec (\downarrow \nu x)((\downarrow \nu * xvec)P'))\ (x \# A_P)\ \Psi_P$

by(*force intro: rBrClose*)

moreover from $Aeq \langle A_P \#* A_{P'} \rangle$ **have** $A_P \#* (p \cdot A_P)$ **by** *simp*
moreover from $Aeq \langle (set\ p \subseteq (set\ (x\#A_P)) \times (set\ A_{P'})) \rangle$
have $(set\ p \subseteq (set\ (x\#A_P)) \times (set\ (p \cdot (x\#A_P))))$ **by** *simp*
moreover from $\langle A_P \#* P \rangle \langle x \# A_P \rangle$ **have** $A_P \#* ((\nu x)P)$ **by** *simp*
moreover from $S \langle x \# A_{P'} \rangle$ **have** $p \cdot x \neq x$
using Aeq **by** *fastforce*
moreover from $\langle x \# A_{P'} \rangle Aeq$ **have** $x \# p \cdot A_P$ **by** *simp*
moreover note $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* (\tau \prec ((\nu x)((\nu*vec)P')) \rangle$
 $\langle A_P \#* C \rangle \langle distinctPerm\ p \rangle$
 $\langle x \# \Psi \rangle \langle x \# C \rangle \langle (p \cdot x) \# A_P \rangle$
ultimately
have $Prop\ C\ \Psi\ ((\nu x)P)\ (\tau \prec ((\nu x)((\nu*vec)P'))\ (p \cdot (x\#A_P))\ (p \cdot \Psi_P)$
by(*fastforce intro!: rAlpha simp add: abs-fresh*)
with $\Psi eq\ Aeq$ **show** *?case* **by** *simp*

next
case(*cOpen* $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_{xP}\ \Psi_{xP}\ C$)
obtain $A_P\ \Psi_P$ **where** $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
and $A_P \#* (\Psi, P, M, xvec, yvec, N, P', A_{xP}, \Psi_{xP}, C, x)$
by(*rule freshFrame*)
then have $A_P \#* \Psi$ **and** $A_P \#* P$ **and** $A_P \#* M$ **and** $A_P \#* xvec$ **and** $A_P \#*$
 $yvec$ **and** $A_P \#* N$ **and** $A_P \#* P'$
and $A_P \#* A_{xP}$ **and** $A_P \#* \Psi_{xP}$ **and** $A_P \#* C$ **and** $x \# A_P$
by *simp+*

from $\langle xvec \#* P \rangle \langle A_P \#* xvec \rangle FrP$ **have** $xvec \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)
from $\langle yvec \#* P \rangle \langle A_P \#* yvec \rangle FrP$ **have** $yvec \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle extractFrame((\nu x)P) = \langle A_{xP}, \Psi_{xP} \rangle \rangle FrP$
have $\langle (x\#A_P), \Psi_P \rangle = \langle A_{xP}, \Psi_{xP} \rangle$
by *simp*
moreover from $\langle x \# A_P \rangle \langle distinct\ A_P \rangle$ **have** *distinct*($x\#A_P$) **by** *simp*
ultimately obtain p **where** $S: set\ p \subseteq set\ (x\#A_P) \times set\ (p \cdot (x\#A_P))$ **and**
distinctPerm p
and $\Psi eq: \Psi_{xP} = p \cdot \Psi_P$ **and** $Aeq: A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using $\langle A_P \#* A_{xP} \rangle \langle x \# A_{xP} \rangle \langle distinct\ A_{xP} \rangle$
apply $-$
apply(*rule frameChainEq'*)
by(*assumption | simp*)+

note $\langle \Psi \triangleright P \mapsto M((\nu*(xvec@yvec))\langle N \rangle \prec P') \rangle FrP \langle distinct\ A_P \rangle$
moreover from $FrP \langle distinct\ A_P \rangle \langle \bigwedge A_P\ \Psi_P\ C. \llbracket extractFrame\ P = \langle A_P, \Psi_P \rangle; \llbracket distinct\ A_P \rrbracket \implies Prop\ C\ \Psi\ P\ (M((\nu*(xvec@yvec))\langle N \rangle \prec P')\ A_P\ \Psi_P) \rangle$
have $\bigwedge C. Prop\ C\ \Psi\ P\ (M((\nu*(xvec@yvec))\langle N \rangle \prec P')\ A_P\ \Psi_P)$ **by** *simp*
moreover note $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \in supp\ N \rangle \langle x \# A_P \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* N \rangle$

$\langle A_P \#* P' \rangle$
 $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* M \rangle \langle xvec \#* \Psi_P \rangle \langle yvec \#* \Psi \rangle \langle yvec \#* P \rangle$
 $\langle yvec \#* M \rangle \langle yvec \#* \Psi_P \rangle$
 $\langle A_P \#* C \rangle \langle x \# C \rangle \langle xvec \#* C \rangle \langle yvec \#* C \rangle \langle xvec \#* yvec \rangle \langle distinct\ xvec \rangle \langle distinct\ yvec \rangle$
ultimately have $Prop\ C\ \Psi\ (\downarrow\nu x)P\ (M(\downarrow\nu*(xvec@x\#yvec))\langle N \rangle \prec P')\ (x\#A_P)$
 Ψ_P
by(metis rOpen)

with $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* N \rangle$
 $\langle A_P \#* P' \rangle \langle A_P \#* A_{xP} \rangle \langle A_P \#* C \rangle \langle x \# A_{xP} \rangle \langle A_P \#* A_{xP} \rangle \langle x \# A_P \rangle$
 $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# C \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle\ Aeq$
 $S\ \langle distinctPerm\ p \rangle$
have $Prop\ C\ \Psi\ (\downarrow\nu x)P\ (M(\downarrow\nu*(xvec@x\#yvec))\langle N \rangle \prec P')\ (p \cdot (x\#A_P))\ (p \cdot \Psi_P)$
apply –
apply(rule rAlpha[**where** $A_P = x\#A_P$])
by(assumption | simp add: abs-fresh fresh-star-def boundOutputFresh)+
with $\Psi eq\ Aeq$ **show** ?case **by**(simp add: eqvts)

next
case(cBrOpen $\Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_{xP}\ \Psi_{xP}\ C$)
obtain $A_P\ \Psi_P$ **where** $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **and** $distinct\ A_P$
and $A_P \#* (\Psi, P, M, xvec, yvec, N, P', A_{xP}, \Psi_{xP}, C, x)$
by(rule freshFrame)

then have $A_P \#* \Psi$ **and** $A_P \#* P$ **and** $A_P \#* M$ **and** $A_P \#* xvec$ **and** $A_P \#* yvec$ **and** $A_P \#* N$ **and** $A_P \#* P'$
and $A_P \#* A_{xP}$ **and** $A_P \#* \Psi_{xP}$ **and** $A_P \#* C$ **and** $x \# A_P$
by simp+

from $\langle xvec \#* P \rangle \langle A_P \#* xvec \rangle\ FrP$ **have** $xvec \#* \Psi_P$
by(force dest: extractFrameFreshChain)

from $\langle yvec \#* P \rangle \langle A_P \#* yvec \rangle\ FrP$ **have** $yvec \#* \Psi_P$
by(force dest: extractFrameFreshChain)

from $\langle extractFrame(\downarrow\nu x)P \rangle = \langle A_{xP}, \Psi_{xP} \rangle\ FrP$
have $\langle (x\#A_P), \Psi_P \rangle = \langle A_{xP}, \Psi_{xP} \rangle$
by simp

moreover from $\langle x \# A_P \rangle \langle distinct\ A_P \rangle$ **have** $distinct(x\#A_P)$ **by** simp
ultimately obtain p **where** $S: set\ p \subseteq set\ (x\#A_P) \times set\ (p \cdot (x\#A_P))$ **and** $distinctPerm\ p$
and $\Psi eq: \Psi_{xP} = p \cdot \Psi_P$ **and** $Aeq: A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using $\langle A_P \#* A_{xP} \rangle \langle x \# A_{xP} \rangle \langle distinct\ A_{xP} \rangle$
apply –
apply(rule frameChainEq')
by(assumption | simp)+

note $\langle \Psi \triangleright P \mapsto \downarrow M(\downarrow\nu*(xvec@yvec))\langle N \rangle \prec P' \rangle\ FrP\ \langle distinct\ A_P \rangle$
moreover from $FrP\ \langle distinct\ A_P \rangle \langle \bigwedge A_P\ \Psi_P\ C. \llbracket extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P \rrbracket \implies Prop\ C\ \Psi\ P\ (\downarrow M(\downarrow\nu*(xvec@yvec))\langle N \rangle \prec P')\ A_P\ \Psi_P$

have $\bigwedge C. Prop C \Psi P (\downarrow M(\nu^*(xvec@yvec))\langle N \rangle \prec P') A_P \Psi_P$ **by** *simp*
moreover note $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \in supp N \rangle \langle x \# A_P \rangle$
 $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle \langle A_P \# M \rangle \langle A_P \# xvec \rangle \langle A_P \# yvec \rangle \langle A_P \# N \rangle$
 $\langle A_P \# P' \rangle$
 $\langle xvec \# \Psi \rangle \langle xvec \# P \rangle \langle xvec \# M \rangle \langle xvec \# \Psi_P \rangle \langle yvec \# \Psi \rangle \langle yvec \# P \rangle$
 $\langle yvec \# M \rangle \langle yvec \# \Psi_P \rangle$
 $\langle A_P \# C \rangle \langle x \# C \rangle \langle xvec \# C \rangle \langle yvec \# C \rangle \langle xvec \# yvec \rangle \langle distinct\ xvec \rangle \langle distinct\ yvec \rangle$
ultimately have $Prop C \Psi (\downarrow \nu x)P (\downarrow M(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P') (x\#A_P)$
 Ψ_P
by(*metis rBrOpen*)

with $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle \langle A_P \# M \rangle \langle A_P \# xvec \rangle \langle A_P \# yvec \rangle \langle A_P \# N \rangle$
 $\langle A_P \# P' \rangle \langle A_P \# A_{xP} \rangle \langle A_P \# C \rangle \langle x \# A_{xP} \rangle \langle A_P \# A_{xP} \rangle \langle x \# A_P \rangle$
 $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# C \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle Aeq$
 $S \langle distinctPerm\ p \rangle$
have $Prop C \Psi (\downarrow \nu x)P (\downarrow M(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P') (p \cdot (x\#A_P)) (p \cdot$
 $\Psi_P)$
apply –
apply(*rule rAlpha[where $A_P = x\#A_P$]*)
by(*assumption | simp add: abs-fresh fresh-star-def boundOutputFresh*) +
with $\Psi eq Aeq$ **show** ?*case by*(*simp add: eqvts*)

next
case(*cScope $\Psi P \alpha P' x A_{xP} \Psi_{xP} C$*)
obtain $A_P \Psi_P$ **where** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** *distinct A_P*
and $A_P \# (\Psi, P, \alpha, P', A_{xP}, \Psi_{xP}, C, x)$
by(*rule freshFrame*)
then have $A_P \# \Psi$ **and** $A_P \# P$ **and** $A_P \# \alpha$ **and** $A_P \# P'$
and $A_P \# A_{xP}$ **and** $A_P \# \Psi_{xP}$ **and** $A_P \# C$ **and** $x \# A_P$
by *simp* +

from $\langle bn\ \alpha \# P \rangle \langle A_P \# \alpha \rangle FrP$ **have** $bn\ \alpha \# \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle extractFrame(\downarrow \nu x)P \rangle = \langle A_{xP}, \Psi_{xP} \rangle FrP$
have $\langle (x\#A_P), \Psi_P \rangle = \langle A_{xP}, \Psi_{xP} \rangle$
by *simp*
moreover from $\langle x \# A_P \rangle \langle distinct\ A_P \rangle$ **have** $distinct(x\#A_P)$ **by** *simp*
ultimately obtain p **where** $S: set\ p \subseteq set\ (x\#A_P) \times set\ (p \cdot (x\#A_P))$ **and**
distinctPerm p
and $\Psi eq: \Psi_{xP} = p \cdot \Psi_P$ **and** $Aeq: A_{xP} = (p \cdot x)\#(p \cdot A_P)$
using $\langle A_P \# A_{xP} \rangle \langle x \# A_{xP} \rangle \langle distinct\ A_{xP} \rangle$
apply –
apply(*rule frameChainEq'*)
by(*assumption | simp*) +

note $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle FrP \langle distinct\ A_P \rangle$
moreover from $FrP \langle distinct\ A_P \rangle \langle \bigwedge A_P \Psi_P C. \llbracket extractFrame P = \langle A_P, \Psi_P \rangle; distinct\ A_P \rrbracket \implies Prop C \Psi P (\alpha \prec P') A_P \Psi_P \rangle$

```

have  $\bigwedge C. Prop C \Psi P (\alpha \prec P') A_P \Psi_P$  by simp
moreover note  $\langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# A_P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle distinct(bn \alpha) \rangle$ 
 $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* \Psi_P \rangle \langle A_P \#* C \rangle \langle x \# C \rangle$ 
 $\langle bn \alpha \#* C \rangle$ 
ultimately have  $Prop C \Psi ((\nu x)P) (\alpha \prec ((\nu x)P')) (x\#A_P) \Psi_P$ 
by(metis rScope)

with  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle A_P \#* A_{xP} \rangle \langle A_P \#* C \rangle \langle x \# A_{xP} \rangle \langle A_P \#* A_{xP} \rangle \langle x \# A_P \rangle$ 
 $\langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# C \rangle$  Aeq
 $S \langle distinctPerm p \rangle$ 
have  $Prop C \Psi ((\nu x)P) (\alpha \prec ((\nu x)P')) (p \cdot (x\#A_P)) (p \cdot \Psi_P)$ 
apply –
apply(rule rAlpha[where A_P=x#A_P])
by(assumption | simp add: abs-fresh fresh-star-def)+
with  $\Psi_{eq} A_{eq}$  show ?case by(simp add: eqvts)
next
case(Bang  $\Psi P Rs A_{bP} \Psi_{bP} C$ )

obtain  $A_P \Psi_P$  where FrP: extractFrame P =  $\langle A_P, \Psi_P \rangle$  and distinct  $A_P$ 
and  $A_P \#* (\Psi, P, Rs, C)$ 
by(rule freshFrame)
then have  $A_P \#* \Psi$  and  $A_P \#* P$  and  $A_P \#* Rs$  and  $A_P \#* C$ 
by simp+

note  $\langle \Psi \triangleright P \parallel !P \mapsto Rs \rangle \langle guarded P \rangle FrP \langle distinct A_P \rangle$ 
moreover from FrP have  $extractFrame (P \parallel !P) = \langle A_P, \Psi_P \otimes \mathbf{1} \rangle$ 
by simp
with  $\langle distinct A_P \rangle \langle \bigwedge A_P \Psi_P C. \llbracket extractFrame (P \parallel !P) = \langle A_P, \Psi_P \rangle; distinct A_P \rrbracket \implies Prop C \Psi (P \parallel !P) Rs A_P \Psi_P \rangle$ 
have  $\bigwedge C. Prop C \Psi (P \parallel !P) Rs A_P (\Psi_P \otimes \mathbf{1})$  by simp
moreover from  $\langle guarded P \rangle FrP$  have  $\Psi_P \simeq \mathbf{1}$  and  $supp \Psi_P = (\{\} :: name set)$ 
by(metis guardedStatEq)+
moreover note  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Rs \rangle \langle A_P \#* C \rangle$ 
ultimately have  $Prop C \Psi (!P) Rs (\parallel) (\mathbf{1})$ 
by(rule rBang)
then show ?case using  $\langle extractFrame(!P) = \langle A_{bP}, \Psi_{bP} \rangle \rangle$  by simp
qed

lemma semanticsFrameInduct'[consumes 5, case-names cAlpha cFrameAlpha cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang]:
fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $Rs$  :: ('a, 'b, 'c) residual
and  $A_P$  :: name list
and  $\Psi_P$  :: 'b
and  $Prop$  :: 'f::fs-name  $\Rightarrow$  'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$  'a action  $\Rightarrow$ 

```

$(\text{'a, 'b, 'c}) \text{ psi} \Rightarrow \text{name list} \Rightarrow \text{'b} \Rightarrow \text{bool}$
and $C :: \text{'f}::\text{fs-name}$

assumes $\text{Trans}: \Psi \triangleright P \mapsto \alpha \prec P'$
and $\text{FrP}: \text{extractFrame } P = \langle A_P, \Psi_P \rangle$
and $\text{distinct } A_P$
and $\text{bn } \alpha \#* \text{subject } \alpha$
and $\text{distinct}(\text{bn } \alpha)$
and $\text{rAlpha}: \bigwedge \Psi P \alpha P' p A_P \Psi_P C. \llbracket \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P; \text{bn } \alpha \#* \text{subject } \alpha; \text{bn } \alpha \#* \Psi_P; \text{bn } \alpha \#* C; \text{bn } \alpha \#* (p \cdot \alpha); A_P \#* \Psi; A_P \#* P; A_P \#* \alpha; A_P \#* P'; A_P \#* C; \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha)); \text{distinctPerm } p; \text{bn}(p \cdot \alpha) \#* \alpha; (\text{bn}(p \cdot \alpha)) \#* P'; \text{Prop } C \Psi P \alpha P' A_P \Psi_P \rrbracket \Longrightarrow$

$\text{Prop } C \Psi P (p \cdot \alpha) (p \cdot P') A_P \Psi_P$
and $\text{rFrameAlpha}: \bigwedge \Psi P A_P \Psi_P p \alpha P' C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* (p \cdot A_P); A_P \#* \alpha; A_P \#* P'; A_P \#* C; \text{set } p \subseteq \text{set } A_P \times \text{set}(p \cdot A_P); \text{distinctPerm } p; A_P \#* \text{subject } \alpha; \text{Prop } C \Psi P \alpha P' A_P \Psi_P \rrbracket \Longrightarrow \text{Prop } C \Psi P \alpha P' (p \cdot A_P) (p \cdot \Psi_P)$

and $\text{rInput}: \bigwedge \Psi M K \text{xvec } N \text{Tvec } P C. \llbracket \Psi \vdash M \leftrightarrow K; \text{distinct } \text{xvec}; \text{set } \text{xvec} \subseteq \text{supp } N; \text{length } \text{xvec} = \text{length } \text{Tvec}; \text{xvec} \#* \Psi; \text{xvec} \#* M; \text{xvec} \#* K; \text{xvec} \#* C \rrbracket \Longrightarrow \text{Prop } C \Psi (M(\lambda \text{xvec}. N)).P$

$(K(\llbracket (N[\text{xvec}::=\text{Tvec}]) \rrbracket)) (P[\text{xvec}::=\text{Tvec}]) (\llbracket \rrbracket) (\mathbf{1})$

and $\text{rBrInput}: \bigwedge \Psi M K \text{xvec } N \text{Tvec } P C. \llbracket \Psi \vdash K \succeq M; \text{distinct } \text{xvec}; \text{set } \text{xvec} \subseteq \text{supp } N; \text{length } \text{xvec} = \text{length } \text{Tvec}; \text{xvec} \#* \Psi; \text{xvec} \#* M; \text{xvec} \#* K; \text{xvec} \#* C \rrbracket \Longrightarrow \text{Prop } C \Psi (M(\lambda \text{xvec}. N)).P$

$(\llbracket K(\llbracket (N[\text{xvec}::=\text{Tvec}]) \rrbracket)) (P[\text{xvec}::=\text{Tvec}]) (\llbracket \rrbracket) (\mathbf{1})$

and $\text{rOutput}: \bigwedge \Psi M K N P C. \Psi \vdash M \leftrightarrow K \Longrightarrow \text{Prop } C \Psi (M\langle N \rangle.P) (K\langle N \rangle) P (\llbracket \rrbracket) (\mathbf{1})$

and $\text{rBrOutput}: \bigwedge \Psi M K N P C. \Psi \vdash M \preceq K \Longrightarrow \text{Prop } C \Psi (M\langle N \rangle.P) (\llbracket K\langle N \rangle \rrbracket) P (\llbracket \rrbracket) (\mathbf{1})$

and $\text{rCase}: \bigwedge \Psi P \alpha P' \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P; \bigwedge C. \text{Prop } C \Psi P \alpha P' A_P \Psi_P; (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P; \Psi_P \simeq \mathbf{1}; (\text{supp } \Psi_P) = (\{\}::\text{name set}); A_P \#* \Psi; A_P \#* P; A_P \#* \alpha; A_P \#* P'; A_P \#* C \rrbracket \Longrightarrow \text{Prop } C \Psi (\text{Cases } Cs) \alpha P' (\llbracket \rrbracket) (\mathbf{1})$

and $\text{rPar1}: \bigwedge \Psi \Psi_Q P \alpha P' A_Q Q A_P \Psi_P C. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q; \rrbracket$

$\wedge C. Prop C (\Psi \otimes \Psi_Q) P \alpha P' A_P \Psi_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* P'; A_P \#* A_Q; A_P$
 $\#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* P'; A_Q \#* \Psi_P;$
 $bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* subject \alpha; bn \alpha \#* \Psi_P;$
 $bn \alpha \#* \Psi_Q;$
 $A_P \#* C; A_Q \#* C; bn \alpha \#* C \implies$
 $Prop C \Psi (P \parallel Q) \alpha (P' \parallel Q) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rPar2: \wedge \Psi \Psi_P Q \alpha Q' A_P P A_Q \Psi_Q C.$
 $\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rrbracket;$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\wedge C. Prop C (\Psi \otimes \Psi_P) Q \alpha Q' A_Q \Psi_Q;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* Q'; A_P \#* A_Q; A_P$
 $\#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* Q'; A_Q \#* \Psi_P;$
 $bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* subject \alpha; bn \alpha \#* \Psi_P;$
 $bn \alpha \#* \Psi_Q;$
 $A_P \#* C; A_Q \#* C; bn \alpha \#* C \implies$
 $Prop C \Psi (P \parallel Q) \alpha (P \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rComm1: \wedge \Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\wedge C. Prop C (\Psi \otimes \Psi_Q) P (M(N)) P' A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec) \langle N \rangle \prec Q'; extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle; distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; distinct xvec;$
 $\wedge C. Prop C (\Psi \otimes \Psi_P) Q (K(\nu * xvec) \langle N \rangle) Q' A_Q \Psi_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#*$
 $M;$
 $xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) (\tau) (\nu * xvec) (P' \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rComm2: \wedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\wedge C. Prop C (\Psi \otimes \Psi_Q) P (M(\nu * xvec) \langle N \rangle) P' A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $\wedge C. Prop C (\Psi \otimes \Psi_P) Q (K(N)) Q' A_Q \Psi_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#*$
 $M;$
 $xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C \implies$

$Prop\ C\ \Psi\ (P\ \parallel\ Q)\ (\tau)\ (\nu^*xvec)\ (P'\ \parallel\ Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and $rBrMerge: \bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(N) \prec P'; \bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P$
 $(iM(N))\ P'\ A_P\ \Psi_P;$
 $extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q'; \bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q$
 $(iM(N))\ Q'\ A_Q\ \Psi_Q;$
 $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle; distinct\ A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C \rrbracket \implies$
 $Prop\ C\ \Psi\ (P\ \parallel\ Q)\ (iM(N))\ (P'\ \parallel\ Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and $rBrComm1: \bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ xvec\ Q'\ A_Q\ C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(N) \prec P'; extractFrame\ P = \langle A_P, \Psi_P \rangle;$
 $distinct\ A_P;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (iM(N))\ P'\ A_P\ \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(\nu^*xvec)\langle N \rangle \prec Q'; extractFrame\ Q = \langle A_Q,$
 $\Psi_Q \rangle; distinct\ A_Q;$
 $distinct\ xvec;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (iM(\nu^*xvec)\langle N \rangle)\ Q'\ A_Q\ \Psi_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M \rrbracket \implies$
 $Prop\ C\ \Psi\ (P\ \parallel\ Q)\ (iM(\nu^*xvec)\langle N \rangle)\ (P'\ \parallel\ Q')\ (A_P @ A_Q)\ (\Psi_P \otimes$
 $\Psi_Q)$
and $rBrComm2: \bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q\ C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu^*xvec)\langle N \rangle \prec P'; extractFrame\ P = \langle A_P,$
 $\Psi_P \rangle; distinct\ A_P;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (iM(\nu^*xvec)\langle N \rangle)\ P'\ A_P\ \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q'; extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct\ A_Q;$
 $\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (iM(N))\ Q'\ A_Q\ \Psi_Q;$
 $distinct\ xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M \rrbracket \implies$
 $Prop\ C\ \Psi\ (P\ \parallel\ Q)\ (iM(\nu^*xvec)\langle N \rangle)\ (P'\ \parallel\ Q')\ (A_P @ A_Q)\ (\Psi_P \otimes$
 $\Psi_Q)$
and $rBrClose: \bigwedge \Psi\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ x\ C.$
 $\llbracket \Psi \triangleright P \mapsto iM(\nu^*xvec)\langle N \rangle \prec P';$
 $x \in supp\ M;$
 $\bigwedge C. Prop\ C\ \Psi\ P\ (iM(\nu^*xvec)\langle N \rangle)\ P'\ A_P\ \Psi_P;$

$extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* xvec;$
 $distinct xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P;$
 $xvec \#* M;$
 $x \# \Psi; x \# xvec; x \# A_P;$
 $A_P \#* C; xvec \#* C; x \# C \implies$
 $Prop C \Psi ((\nu x)P) (\tau) ((\nu x)((\nu * xvec)P')) (x \# A_P) \Psi_P$
and $rOpen: \bigwedge \Psi P M xvec yvec N P' x A_P \Psi_P y C.$
 $\llbracket \Psi \triangleright P \mapsto M(\nu*(xvec@yvec))\langle N \rangle \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi P (M(\nu*(xvec@yvec))\langle N \rangle) P' A_P \Psi_P; x \in supp$
 $N; x \# \Psi; x \# M;$
 $x \# A_P; x \# xvec; x \# yvec; A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#*$
 $N; A_P \#* P';$
 $A_P \#* xvec; A_P \#* yvec; xvec \#* yvec; distinct xvec; distinct yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* \Psi_P;$
 $yvec \#* \Psi; yvec \#* P; yvec \#* M; A_P \#* C; x \# C; xvec \#* C; yvec$
 $\#* C;$
 $y \neq x; y \# \Psi; y \# P; y \# M; y \# xvec; y \# yvec; y \# N; y \# P'; y \#$
 $A_P; y \# \Psi_P; y \# C \implies$
 $Prop C \Psi ((\nu x)P) (M(\nu*(xvec@y\#yvec))\langle [(x, y)] \cdot N \rangle) ([(x,$
 $y)] \cdot P') (x \# A_P) \Psi_P$
and $rBrOpen: \bigwedge \Psi P M xvec yvec N P' x A_P \Psi_P y C.$
 $\llbracket \Psi \triangleright P \mapsto {}_i M(\nu*(xvec@yvec))\langle N \rangle \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi P ({}_i M(\nu*(xvec@yvec))\langle N \rangle) P' A_P \Psi_P; x \in supp$
 $N; x \# \Psi; x \# M;$
 $x \# A_P; x \# xvec; x \# yvec; A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#*$
 $N; A_P \#* P';$
 $A_P \#* xvec; A_P \#* yvec; xvec \#* yvec; distinct xvec; distinct yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* \Psi_P;$
 $yvec \#* \Psi; yvec \#* P; yvec \#* M; A_P \#* C; x \# C; xvec \#* C; yvec$
 $\#* C;$
 $y \neq x; y \# \Psi; y \# P; y \# M; y \# xvec; y \# yvec; y \# N; y \# P'; y \#$
 $A_P; y \# \Psi_P; y \# C \implies$
 $Prop C \Psi ((\nu x)P) ({}_i M(\nu*(xvec@y\#yvec))\langle [(x, y)] \cdot N \rangle) ([(x,$
 $y)] \cdot P') (x \# A_P) \Psi_P$
and $rScope: \bigwedge \Psi P \alpha P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto \alpha \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi P \alpha P' A_P \Psi_P;$
 $x \# \Psi; x \# \alpha; x \# A_P; A_P \#* \Psi; A_P \#* P;$
 $A_P \#* \alpha; A_P \#* P';$
 $bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* subject \alpha; bn \alpha \#* \Psi_P;$
 $A_P \#* C; x \# C; bn \alpha \#* C \implies$
 $Prop C \Psi ((\nu x)P) \alpha ((\nu x)P') (x \# A_P) \Psi_P$
and $rBang: \bigwedge \Psi P \alpha P' A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto \alpha \prec P'; guarded P; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\bigwedge C. Prop C \Psi (P \parallel !P) \alpha P' A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; supp \Psi_P$


```

= ({}::name set);
       $A_P \#* \Psi; A_P \#* P; A_P \#* \alpha; A_P \#* P'; A_P \#* C \parallel \implies Prop C \Psi$ 
(!P)  $\alpha P' (\parallel)$  (1)
shows  $Prop C \Psi P \alpha P' A_P \Psi_P$ 
  using  $Trans FrP \langle distinct A_P \rangle \langle bn \alpha \#* subject \alpha \rangle \langle distinct(bn \alpha) \rangle$ 
proof(nominal-induct  $\Psi P Rs==\alpha \prec P' A_P \Psi_P$  avoiding:  $C \alpha P'$  rule: semanticsFrameInduct)
  case cAlpha
  then show ?case using rFrameAlpha
  by auto
next
  case cInput
  then show ?case using rInput
  by(auto simp add: residualInject)
next
  case cBrInput
  then show ?case using rBrInput
  by(auto simp add: residualInject)
next
  case cOutput
  then show ?case using rOutput
  by(auto simp add: residualInject)
next
  case cBrOutput
  then show ?case using rBrOutput
  by(auto simp add: residualInject)
next
  case cCase
  then show ?case using rCase
  by(auto simp add: residualInject)
next
  case(cPar1  $\Psi \Psi_Q P \alpha P' A_Q Q A_P \Psi_P C \alpha' P''$ )
  note  $\langle \alpha \prec (P' \parallel Q) = \alpha' \prec P'' \rangle$ 
  moreover from  $\langle bn \alpha \#* \alpha' \rangle$  have  $bn \alpha \#* (bn \alpha')$  by auto
  moreover note  $\langle distinct (bn \alpha) \rangle \langle distinct (bn \alpha') \rangle$ 
  moreover from  $\langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha' \#* subject \alpha' \rangle$ 
  have  $bn \alpha \#* (\alpha \prec P' \parallel Q)$  and  $bn \alpha' \#* (\alpha' \prec P'')$  by simp+
  ultimately obtain  $p$  where  $S: (set p) \subseteq (set (bn \alpha)) \times (set (bn(p \cdot \alpha)))$  and
distinctPerm p
  and  $\alpha Eq: \alpha' = p \cdot \alpha$  and  $P' eq: P'' = p \cdot (P' \parallel Q)$  and  $(bn(p \cdot \alpha)) \#* \alpha$ 
  and  $(bn(p \cdot \alpha)) \#* (P' \parallel Q)$ 
  by(rule residualEq)

  note  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle$   $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$   $\langle distinct A_Q \rangle$ 
  moreover from  $\langle bn \alpha \#* subject \alpha \rangle \langle distinct (bn \alpha) \rangle \langle A_P \#* \alpha \rangle$ 
  have  $\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P \alpha P' A_P \Psi_P$  by(fastforce intro: cPar1)

  moreover note  $\langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \alpha \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* C \rangle$ 
 $\langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle A_P \#* C \rangle$ 

```

$\langle bn \ \alpha \ \#* \ Q \rangle \langle distinct(bn \ \alpha) \rangle \langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ \Psi_Q \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle$
 $\langle extractFrame \ P = \langle A_P, \Psi_P \rangle \rangle \langle distinct \ A_P \rangle \langle A_P \ \#* \ A_Q \rangle \langle A_P \ \#* \ \Psi_Q \rangle \langle A_Q \ \#* \ \Psi_P \rangle \langle bn \ \alpha \ \#* \ \Psi_P \rangle$
ultimately have $Prop \ C \ \Psi \ (P \parallel Q) \ \alpha \ (P' \parallel Q) \ (A_P @ A_Q) \ (\Psi_P \otimes \Psi_Q)$
by(metis rPar1)
with $\langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle \langle bn \ \alpha \ \#* \ (bn \ \alpha') \rangle \ S \ \langle distinctPerm \ p \rangle \langle bn(p \cdot \alpha) \ \#* \ \alpha \rangle \langle bn(p \cdot \alpha) \ \#* \ (P' \parallel Q) \rangle \langle bn \ \alpha \ \#* \ \Psi_P \rangle \langle bn \ \alpha \ \#* \ \Psi_Q \rangle \langle A_P \ \#* \ \alpha \rangle \langle A_Q \ \#* \ \alpha \rangle \langle A_P \ \#* \ \alpha' \rangle \langle A_Q \ \#* \ \alpha' \rangle \ \alpha Eq \ \langle bn \ \alpha \ \#* \ \Psi_P \rangle \langle bn \ \alpha \ \#* \ \alpha' \rangle \langle A_P \ \#* \ \Psi \rangle \langle A_Q \ \#* \ \Psi \rangle \langle A_P \ \#* \ P \rangle \langle A_Q \ \#* \ P \rangle \langle A_P \ \#* \ Q \rangle \langle A_Q \ \#* \ Q \rangle \langle A_P \ \#* \ P' \rangle \langle A_Q \ \#* \ P' \rangle \langle A_P \ \#* \ C \rangle \langle A_Q \ \#* \ C \rangle$
have $Prop \ C \ \Psi \ (P \parallel Q) \ (p \cdot \alpha) \ (p \cdot (P' \parallel Q)) \ (A_P @ A_Q) \ (\Psi_P \otimes \Psi_Q)$
by(fastforce intro!: rAlpha)
with $\alpha Eq \ P' eq \ \langle distinctPerm \ p \rangle$ **show** ?case **by** simp
next
case(cPar2 $\Psi \ \Psi_P \ Q \ \alpha \ Q' \ A_P \ P \ A_Q \ \Psi_Q \ C \ \alpha' \ Q''$)
note $\langle \alpha \prec (P \parallel Q') = \alpha' \prec Q'' \rangle$
moreover from $\langle bn \ \alpha \ \#* \ \alpha' \rangle$ **have** $bn \ \alpha \ \#* \ (bn \ \alpha')$ **by** auto
moreover note $\langle distinct \ (bn \ \alpha) \rangle \langle distinct \ (bn \ \alpha') \rangle$
moreover from $\langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha' \ \#* \ subject \ \alpha' \rangle$
have $bn \ \alpha \ \#* \ (\alpha \prec P \parallel Q')$ **and** $bn \ \alpha' \ \#* \ (\alpha' \prec Q'')$ **by** simp+
ultimately obtain p **where** $S: (set \ p) \subseteq (set \ (bn \ \alpha)) \times (set \ (bn(p \cdot \alpha)))$ **and** $distinctPerm \ p$
and $\alpha Eq: \alpha' = p \cdot \alpha$ **and** $Q' eq: Q'' = p \cdot (P \parallel Q')$ **and** $(bn(p \cdot \alpha)) \ \#* \ \alpha$
and $(bn(p \cdot \alpha)) \ \#* \ (P \parallel Q')$
by(rule residualEq)

note $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle \langle extractFrame \ P = \langle A_P, \Psi_P \rangle \rangle \langle distinct \ A_P \rangle$
moreover from $\langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle distinct \ (bn \ \alpha) \rangle \langle A_Q \ \#* \ \alpha \rangle$
have $\bigwedge C. Prop \ C \ (\Psi \otimes \Psi_P) \ Q \ \alpha \ Q' \ A_Q \ \Psi_Q$ **by**(fastforce intro!: cPar2)

moreover note $\langle A_Q \ \#* \ P \rangle \langle A_Q \ \#* \ Q \rangle \langle A_Q \ \#* \ \Psi \rangle \langle A_Q \ \#* \ \alpha \rangle \langle A_Q \ \#* \ Q' \rangle \langle A_Q \ \#* \ C \rangle \langle A_P \ \#* \ P \rangle \langle A_P \ \#* \ Q \rangle \langle A_P \ \#* \ \Psi \rangle \langle A_P \ \#* \ \alpha \rangle \langle A_P \ \#* \ Q' \rangle \langle A_P \ \#* \ C \rangle$
 $\langle bn \ \alpha \ \#* \ Q \rangle \langle distinct(bn \ \alpha) \rangle \langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ \Psi_Q \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle$
 $\langle extractFrame \ Q = \langle A_Q, \Psi_Q \rangle \rangle \langle distinct \ A_Q \rangle \langle A_P \ \#* \ A_Q \rangle \langle A_P \ \#* \ \Psi_Q \rangle \langle A_Q \ \#* \ \Psi_P \rangle \langle bn \ \alpha \ \#* \ \Psi_P \rangle$
ultimately have $Prop \ C \ \Psi \ (P \parallel Q) \ \alpha \ (P \parallel Q') \ (A_P @ A_Q) \ (\Psi_P \otimes \Psi_Q)$
by(fastforce intro!: rPar2)
with $\langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle bn \ \alpha \ \#* \ C \rangle \langle bn \ \alpha \ \#* \ (bn \ \alpha') \rangle \ S \ \langle distinctPerm \ p \rangle \langle bn(p \cdot \alpha) \ \#* \ \alpha \rangle \langle bn(p \cdot \alpha) \ \#* \ (P \parallel Q') \rangle \langle bn \ \alpha \ \#* \ \Psi_P \rangle \langle bn \ \alpha \ \#* \ \Psi_Q \rangle \langle A_P \ \#* \ \alpha \rangle \langle A_Q \ \#* \ \alpha \rangle \langle A_P \ \#* \ \alpha' \rangle \langle A_Q \ \#* \ \alpha' \rangle \ \alpha Eq \ \langle bn \ \alpha \ \#* \ \alpha' \rangle \langle A_P \ \#* \ \Psi \rangle \langle A_Q \ \#* \ \Psi \rangle \langle A_P \ \#* \ P \rangle \langle A_Q \ \#* \ P \rangle \langle A_P \ \#* \ Q \rangle \langle A_Q \ \#* \ Q \rangle \langle A_P \ \#* \ Q' \rangle \langle A_Q \ \#* \ Q' \rangle \langle A_P \ \#* \ C \rangle \langle A_Q \ \#* \ C \rangle$
have $Prop \ C \ \Psi \ (P \parallel Q) \ (p \cdot \alpha) \ (p \cdot (P \parallel Q')) \ (A_P @ A_Q) \ (\Psi_P \otimes \Psi_Q)$
by(fastforce intro!: rAlpha)
with $\alpha Eq \ Q' eq \ \langle distinctPerm \ p \rangle$ **show** ?case **by** simp
next
case(cComm1 $\Psi \ \Psi_Q \ P \ M \ N \ P' \ A_P \ \Psi_P \ Q \ K \ xvec \ Q' \ A_Q \ C \ \alpha \ P''$)

```

then show ?case using rComm1
  apply –
  apply(drule meta-spec[where  $x=M(\downarrow N)$ ])
  apply(drule meta-spec[where  $x=K(\downarrow \nu * xvec)$ ]( $N$ ))
  by(auto simp add: residualInject)
next
case(cComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q C \alpha Q'$ )
then show ?case using rComm2
  apply –
  apply(drule meta-spec[where  $x=M(\downarrow \nu * xvec)$ ]( $N$ ))
  apply(drule meta-spec[where  $x=K(\downarrow N)$ ])
  by(auto simp add: residualInject)
next
case(cBrMerge  $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C \alpha P''$ )
then show ?case using rBrMerge
  apply –
  apply(drule meta-spec[where  $x=\downarrow M(\downarrow N)$ ])
  apply(drule meta-spec[where  $x=\downarrow M(\downarrow N)$ ])
  by(auto simp add: residualInject)
next
case(cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C \alpha P''$ )
have bn ( $\downarrow M(\downarrow N)$ )  $\#*$  subject ( $\downarrow M(\downarrow N)$ ) by simp
moreover have distinct (bn ( $\downarrow M(\downarrow N)$ )) by simp
moreover have  $\downarrow M(\downarrow N) \prec P' = \downarrow M(\downarrow N) \prec P'$  by simp
moreover note cBrComm1
ultimately have inProp:  $\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P (\downarrow M(\downarrow N)) P' A_P \Psi_P$  by
simp

note  $\langle xvec \#* M \rangle \langle distinct xvec \rangle$  cBrComm1
then have outProp:  $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (\downarrow M(\downarrow \nu * xvec)(N)) Q' A_Q \Psi_Q$  by
simp

note inProp outProp cBrComm1
then have bigProp: Prop C  $\Psi (P \parallel Q) (\downarrow M(\downarrow \nu * xvec)(N)) (P' \parallel Q') (A_P @ A_Q)$ 
 $(\Psi_P \otimes \Psi_Q)$  by (simp add: rBrComm1)

note  $\langle \downarrow M(\downarrow \nu * xvec)(N) \prec (P' \parallel Q') = \alpha \prec P'' \rangle$ 
moreover from  $\langle xvec \#* \alpha \rangle$  have bn ( $\downarrow M(\downarrow \nu * xvec)(N)$ )  $\#*$  (bn  $\alpha$ ) by simp
moreover from  $\langle distinct xvec \rangle$  have distinct (bn ( $\downarrow M(\downarrow \nu * xvec)(N)$ )) by simp
moreover note  $\langle distinct (bn \alpha) \rangle$ 
moreover from  $\langle xvec \#* M \rangle \langle bn \alpha \#* subject \alpha \rangle$ 
have bn ( $\downarrow M(\downarrow \nu * xvec)(N)$ )  $\#*$  ( $\downarrow M(\downarrow \nu * xvec)(N) \prec P' \parallel Q'$ ) and bn  $\alpha \#*$  ( $\alpha \prec P''$ ) by simp+
ultimately obtain p where S: (set p)  $\subseteq$  (set (bn ( $\downarrow M(\downarrow \nu * xvec)(N)$ )))  $\times$  (set (bn (p
 $\cdot (\downarrow M(\downarrow \nu * xvec)(N))$ ))) and distinctPerm p
and  $\alpha Eq$ :  $\alpha = p \cdot (\downarrow M(\downarrow \nu * xvec)(N))$  and P'eq:  $P'' = p \cdot (P' \parallel Q')$ 
and (bn (p  $\cdot (\downarrow M(\downarrow \nu * xvec)(N))$ ))  $\#*$  ( $\downarrow M(\downarrow \nu * xvec)(N)$ ) and (bn (p  $\cdot (\downarrow M(\downarrow \nu * xvec)(N))$ ))
 $\#*$  ( $P' \parallel Q'$ )
by(rule residualEq) simp

```

from $\langle xvec \#* \Psi \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* \Psi$ **by simp**
moreover from $\langle xvec \#* P \rangle \langle xvec \#* Q \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (P \parallel Q)$
by simp
moreover note $\langle xvec \#* M \rangle$
moreover from $\langle xvec \#* \Psi_P \rangle \langle xvec \#* \Psi_Q \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (\Psi_P$
 $\otimes \Psi_Q)$ **by auto**
moreover from $\langle xvec \#* C \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* C$ **by simp**
moreover from $\langle xvec \#* \alpha \rangle \alpha Eq$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (p \cdot (iM(\nu*xvec)\langle N \rangle))$
by simp
moreover from $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $(A_P @ A_Q) \#* \Psi$ **by simp**
moreover from $\langle A_P \#* P \rangle \langle A_Q \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* P \rangle$ **have** $(A_P @ A_Q)$
 $\#* (P \parallel Q)$ **by simp**
moreover from $\langle A_P \#* \alpha \rangle \langle A_Q \#* \alpha \rangle$ **have** $(A_P @ A_Q) \#* \alpha$ **by simp**
moreover from $\langle A_P \#* P' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* P' \rangle$ **have** $(A_P @$
 $A_Q) \#* (P' \parallel Q')$ **by simp**
moreover from $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle$ **have** $(A_P @ A_Q) \#* C$ **by simp**
moreover note $S \langle distinctPerm p \rangle \langle (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (iM(\nu*xvec)\langle N \rangle) \rangle$
 $\langle (bn(p \cdot (iM(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q') \rangle$ **bigProp**
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* xvec \rangle \langle A_Q \#* xvec \rangle \langle A_P \#* N \rangle \langle A_Q \#* N \rangle$

ultimately have $Prop C \Psi (P \parallel Q) (p \cdot (iM(\nu*xvec)\langle N \rangle)) (p \cdot (P' \parallel Q')) (A_P$
 $@ A_Q) (\Psi_P \otimes \Psi_Q)$
by $(fastforce intro!: rAlpha)$
then show $?case$ **using** $\alpha Eq P'eq$ **by simp**
next
case $(cBrComm2 \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C \alpha Q')$
have $bn (iM(\nu*xvec)\langle N \rangle) \#* subject (iM(\nu*xvec)\langle N \rangle)$ **by simp**
moreover have $distinct (bn (iM(\nu*xvec)\langle N \rangle))$ **by simp**
moreover have $iM(\nu*xvec)\langle N \rangle \prec Q' = iM(\nu*xvec)\langle N \rangle \prec Q'$ **by simp**
moreover note $cBrComm2$
ultimately have $inProp: \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q (iM(\nu*xvec)\langle N \rangle) Q' A_Q \Psi_Q$ **by**
 $simp$

from $\langle xvec \#* M \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* subject (iM(\nu*xvec)\langle N \rangle)$ **by**
 $simp$
moreover from $\langle distinct xvec \rangle$ **have** $distinct (bn (iM(\nu*xvec)\langle N \rangle))$ **by simp**
moreover have $iM(\nu*xvec)\langle N \rangle \prec P' = iM(\nu*xvec)\langle N \rangle \prec P'$ **by simp**
moreover note $cBrComm2$
ultimately have $outProp: \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P (iM(\nu*xvec)\langle N \rangle) P' A_P$
 Ψ_P **by simp**

note $inProp outProp cBrComm2$
then have $bigProp: Prop C \Psi (P \parallel Q) (iM(\nu*xvec)\langle N \rangle) (P' \parallel Q') (A_P @ A_Q)$
 $(\Psi_P \otimes \Psi_Q)$ **by** $(simp add: rBrComm2)$

note $iM(\nu*xvec)\langle N \rangle \prec (P' \parallel Q') = \alpha \prec Q''$
moreover from $\langle xvec \#* \alpha \rangle$ **have** $bn (iM(\nu*xvec)\langle N \rangle) \#* (bn \alpha)$ **by simp**
moreover note $\langle distinct (bn (iM(\nu*xvec)\langle N \rangle)) \rangle \langle distinct (bn \alpha) \rangle$

moreover from $\langle \text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* \text{ subject } (\text{iM}(\nu*xvec)\langle N \rangle) \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$
have $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* (\text{iM}(\nu*xvec)\langle N \rangle \prec P' \parallel Q')$ **and** $\text{bn } \alpha \#* (\alpha \prec Q')$ **by** *simp+*
ultimately obtain p **where** $S: (\text{set } p) \subseteq (\text{set}(\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle))) \times (\text{set}(\text{bn}(p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))))$ **and** *distinctPerm* p
and $\alpha \text{Eq}: \alpha = p \cdot (\text{iM}(\nu*xvec)\langle N \rangle)$ **and** $P' \text{eq}: Q'' = p \cdot (P' \parallel Q')$ **and** $(\text{bn}(p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))) \#* (\text{iM}(\nu*xvec)\langle N \rangle)$
and $(\text{bn}(p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q')$
by(*rule residualEq*)

from $\langle xvec \#* \Psi \rangle$ **have** $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* \Psi$ **by** *simp*
moreover from $\langle xvec \#* P \rangle \langle xvec \#* Q \rangle$ **have** $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* (P \parallel Q)$
by *simp*
moreover note $\langle \text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* \text{ subject } (\text{iM}(\nu*xvec)\langle N \rangle) \rangle$
moreover from $\langle xvec \#* \Psi_P \rangle \langle xvec \#* \Psi_Q \rangle$ **have** $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* (\Psi_P \otimes \Psi_Q)$ **by** *auto*
moreover from $\langle xvec \#* C \rangle$ **have** $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* C$ **by** *simp*
moreover from $\langle xvec \#* \alpha \rangle \alpha \text{Eq}$ **have** $\text{bn } (\text{iM}(\nu*xvec)\langle N \rangle) \#* (p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))$
by *simp*
moreover from $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $(A_P @ A_Q) \#* \Psi$ **by** *simp*
moreover from $\langle A_P \#* P \rangle \langle A_Q \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* P \rangle$ **have** $(A_P @ A_Q) \#* (P \parallel Q)$ **by** *simp*
moreover from $\langle A_P \#* \alpha \rangle \langle A_Q \#* \alpha \rangle$ **have** $(A_P @ A_Q) \#* \alpha$ **by** *simp*
moreover from $\langle A_P \#* P' \rangle \langle A_Q \#* Q' \rangle \langle A_P \#* Q' \rangle \langle A_Q \#* P' \rangle$ **have** $(A_P @ A_Q) \#* (P' \parallel Q')$ **by** *simp*
moreover from $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle$ **have** $(A_P @ A_Q) \#* C$ **by** *simp*
moreover note $S \langle \text{distinctPerm } p \rangle \langle (\text{bn}(p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))) \#* (\text{iM}(\nu*xvec)\langle N \rangle) \rangle$
 $\langle (\text{bn}(p \cdot (\text{iM}(\nu*xvec)\langle N \rangle))) \#* (P' \parallel Q') \rangle$ *bigProp*
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* xvec \rangle \langle A_Q \#* xvec \rangle \langle A_P \#* N \rangle \langle A_Q \#* N \rangle$

ultimately have *Prop* $C \Psi (P \parallel Q) (p \cdot (\text{iM}(\nu*xvec)\langle N \rangle)) (p \cdot (P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
by(*fastforce intro!: rAlpha*)
then show *?case* **using** $\alpha \text{Eq } P' \text{eq}$ **by** *simp*
next
case(*cBrClose* $\Psi P M xvec N P' A_P \Psi_P x C \alpha P''$)
note $\langle \tau \prec (\nu x)((\nu*xvec)P') = \alpha \prec P'' \rangle$
moreover have $\text{bn } (\tau) \#* (\text{bn } \alpha)$ **by** *simp*
moreover have *distinct* $(\text{bn } (\tau))$ **by** *simp*
moreover note $\langle \text{distinct } (\text{bn } \alpha) \rangle$
moreover have $(\text{bn } (\tau) \#* (\tau \prec (\nu x)((\nu*xvec)P')))$ **by** *simp*
moreover from $\langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$ **have** $\text{bn } \alpha \#* (\alpha \prec P'')$ **by** *simp*
ultimately obtain p **where** $S: (\text{set } p) \subseteq (\text{set}(\text{bn } (\tau))) \times (\text{set}(\text{bn}(p \cdot (\tau))))$
and $\alpha \text{Eq}: \alpha = p \cdot (\tau)$ **and** $P' \text{eq}: P'' = p \cdot ((\nu x)((\nu*xvec)P'))$
and $\text{bn } (\tau) \#* \alpha$ **and** $\text{bn } (\tau) \#* P''$
and $(\text{bn}(p \cdot (\tau))) \#* (\tau)$ **and** $(\text{bn}(p \cdot (\tau))) \#* ((\nu x)((\nu*xvec)P'))$
by(*rule residualEq*) *simp*
moreover from *cBrClose* **have** $\bigwedge C. \text{Prop } C \Psi P (\text{iM}(\nu*xvec)\langle N \rangle) P' A_P \Psi_P$

by *simp*
moreover with *cBrClose* **have** $\text{Prop } C \Psi (\nu x)P (\tau) (\nu x)(\nu * xvec)P'$
 $(x \# A_P) \Psi_P$
by (*simp add: rBrClose*)
with *S* **have** $\text{Prop } C \Psi (\nu x)P (p \cdot \tau) (p \cdot (\nu x)(\nu * xvec)P')$ $(x \# A_P) \Psi_P$ **by**
simp
then show *?case* **using** $\alpha Eq P' eq$
by *simp*
next
case (*cOpen* $\Psi P M xvec yvec N P' x A_P \Psi_P C \alpha P'$)
note $\langle M(\nu*(xvec@x\#yvec)) \rangle \langle N \rangle \prec P' = \alpha \prec P''$
moreover from $\langle xvec \#* \alpha \rangle \langle x \# \alpha \rangle \langle yvec \#* \alpha \rangle$ **have** $(xvec@x\#yvec) \#* (bn \alpha)$
by *auto*
moreover from $\langle xvec \#* yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle distinct \ xvec \rangle \langle distinct \ yvec \rangle$
have $distinct(xvec@x\#yvec)$
by (*auto simp add: fresh-star-def*) (*simp add: fresh-def name-list-supp*)
moreover note $\langle distinct(bn \alpha) \rangle$
moreover from $\langle xvec \#* M \rangle \langle x \# M \rangle \langle yvec \#* M \rangle$ **have** $(xvec@x\#yvec) \#* M$
by *auto*
then have $(xvec@x\#yvec) \#* (M(\nu*(xvec@x\#yvec)) \langle N \rangle \prec P')$ **by** *auto*
moreover from $\langle bn \alpha \#* subject \ \alpha \rangle$ **have** $bn \alpha \#* (\alpha \prec P')$ **by** *simp*
ultimately obtain *p* **where** $S: (set \ p) \subseteq (set(xvec@x\#yvec)) \times (set(p \cdot (xvec@x\#yvec)))$
and $distinctPerm \ p$
and $\alpha eq: \alpha = (p \cdot M)(\nu*(p \cdot (xvec@x\#yvec))) \langle (p \cdot N) \rangle$ **and** $P' eq: P'' = (p \cdot P')$
and $A: (xvec@x\#yvec) \#* ((p \cdot M)(\nu*(p \cdot (xvec@x\#yvec))) \langle (p \cdot N) \rangle)$
and $B: (p \cdot (xvec@x\#yvec)) \#* (M(\nu*(xvec@x\#yvec)) \langle N \rangle)$
and $C: (p \cdot (xvec@x\#yvec)) \#* P'$
apply –
apply (*rule residualEq*)
by (*assumption | simp*)**+**

note $\langle \Psi \triangleright P \mapsto M(\nu*(xvec@yvec)) \langle N \rangle \prec P' \rangle \langle x \in (supp \ N) \rangle$

moreover {
fix *C*
from $\langle xvec \#* M \rangle \langle yvec \#* M \rangle$ **have** $(xvec@yvec) \#* M$ **by** *simp*
moreover from $\langle distinct \ xvec \rangle \langle distinct \ yvec \rangle \langle xvec \#* yvec \rangle$ **have** $distinct(xvec@yvec)$
by (*auto simp add: fresh-star-def name-list-supp fresh-def*)
ultimately have $\text{Prop } C \Psi P (M(\nu*(xvec@yvec)) \langle N \rangle) P' A_P \Psi_P$ **using** $\langle A_P \#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$
by (*fastforce intro!: cOpen*)
}
moreover obtain *y::name* **where** $y \# \Psi$ **and** $y \neq x$ **and** $y \# P$ **and** $y \# xvec$
and $y \# yvec$ **and** $y \# \alpha$ **and** $y \# P'$ **and** $y \# A_P$ **and** $y \# \Psi_P$ **and** $y \# M$ **and** $y \# N$
and $y \# C$ **and** $y \# p$
by (*generate-fresh name*) *auto*
moreover note $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$
 $\langle xvec \#* M \rangle$

$\langle yvec \#* \Psi \rangle \langle yvec \#* P \rangle \langle yvec \#* M \rangle \langle yvec \#* C \rangle \langle x \# C \rangle \langle xvec \#* C \rangle \langle distinct \ xvec \rangle \langle distinct \ yvec \rangle$
 $\langle extractFrame \ P = \langle A_P, \Psi_P \rangle \rangle \langle distinct \ A_P \rangle \langle x \# A_P \rangle \langle xvec \#* yvec \rangle \langle xvec \#* \Psi_P \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* N \rangle \langle A_P \#* P' \rangle \langle A_P \#* C \rangle$
ultimately have $Prop \ C \ \Psi \ (\nu x)P \ (M(\nu*(xvec@y\#yvec)))\langle\langle[(x, y)] \cdot N\rangle\rangle \langle\langle[(x, y)] \cdot P'\rangle\rangle \ (x\#A_P) \ \Psi_P$
by(metis rOpen)
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot M\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot M\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \# M \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# M \rangle$ **have** $D: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle M \rangle\rangle = p \cdot M$
by(auto simp add: eqvts freshChainSimps)
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot xvec\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot xvec\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \# xvec \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# xvec \rangle$ **have** $E: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle xvec \rangle\rangle = p \cdot xvec$
by(auto simp add: eqvts freshChainSimps)
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot yvec\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot yvec\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \# yvec \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# yvec \rangle$ **have** $F: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle yvec \rangle\rangle = p \cdot yvec$
by(auto simp add: eqvts freshChainSimps)
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot x\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot x\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \neq x \rangle \langle y \# p \rangle$ **have** $G: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle y \rangle\rangle = p \cdot x$
apply(simp add: freshChainSimps calc-atm)
apply(subgoal-tac $y \neq p \cdot x$)
apply(clarsimp)
using $A \ \alpha eq$
apply(simp add: eqvts)
apply(subst fresh-atm[symmetric])
apply(simp only: freshChainSimps)
by simp
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot N\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot N\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \# N \rangle \langle x \# \alpha \rangle \langle y \# p \rangle \alpha eq$ **have** $H: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot N\rangle\rangle = p \cdot N$
by(auto simp add: eqvts freshChainSimps)
moreover have $\langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot P'\rangle\rangle = \langle\langle[(x, y)] \cdot p \cdot P'\rangle\rangle$
by(subst perm-compose[symmetric]) simp
with $\langle y \# P' \rangle \langle x \# P'' \rangle \langle y \# p \rangle P'eq$ **have** $I: \langle\langle[(x, y)] \cdot p\rangle\rangle \cdot \langle\langle[(x, y)] \cdot P'\rangle\rangle = p \cdot P'$
by(auto simp add: eqvts freshChainSimps)
from $\langle y \# p \rangle \langle y \neq x \rangle$ **have** $y \neq p \cdot x$
apply(subst fresh-atm[symmetric])
apply(simp only: freshChainSimps)
by simp
moreover from S **have** $\langle\langle[(x, y)] \cdot set \ p\rangle\rangle \subseteq \langle\langle[(x, y)] \cdot (set(xvec@x\#yvec) \times set(p \cdot (xvec@x\#yvec)))\rangle\rangle$

```

by(simp)
with  $\langle y \neq p \cdot x \rangle \langle ([x, y] \cdot p) \cdot y = p \cdot x \rangle \langle x \# \text{xvec} \rangle \langle y \# \text{xvec} \rangle \langle x \# \text{yvec} \rangle \langle y \# \text{yvec} \rangle \langle y \# p \rangle \langle x \# \alpha \rangle \alpha \text{eq}$  have
   $\text{set}([x, y] \cdot p) \subseteq \text{set}(\text{xvec}@y\#\text{yvec}) \times \text{set}([x, y] \cdot p) \cdot (\text{xvec}@y\#\text{yvec})$ 
by(simp add: eqvts calc-atm perm-compose)
moreover note  $\langle \text{xvec} \#* \Psi \rangle \langle \text{yvec} \#* \Psi \rangle \langle \text{xvec} \#* P \rangle \langle \text{yvec} \#* P \rangle \langle \text{xvec} \#* M \rangle \langle \text{yvec} \#* M \rangle$ 
 $\langle \text{yvec} \#* C \rangle$   $S$   $\langle \text{distinctPerm } p \rangle \langle x \# C \rangle \langle \text{xvec} \#* C \rangle \langle \text{xvec} \#* \Psi_P \rangle \langle \text{yvec} \#* \Psi_P \rangle \langle x \# \Psi \rangle$ 
 $\langle A_P \#* \text{xvec} \rangle \langle x \# A_P \rangle \langle A_P \#* \text{yvec} \rangle \langle A_P \#* M \rangle \langle x \# \text{xvec} \rangle \langle x \# \text{yvec} \rangle \langle x \# M \rangle \langle x \# A_P \rangle \langle A_P \#* N \rangle$ 
 $A B C \alpha \text{eq} \langle A_P \#* \alpha \rangle \langle y \# \Psi \rangle \langle y \neq x \rangle \langle y \# P \rangle \langle y \# M \rangle \langle y \# \Psi_P \rangle \langle y \# C \rangle \langle \text{xvec} \#* \alpha \rangle \langle x \# \alpha \rangle \langle \text{yvec} \#* \alpha \rangle \langle y \# \alpha \rangle \langle A_P \#* P \rangle \langle A_P \#* \Psi \rangle \langle y \# A_P \rangle \langle y \# N \rangle \langle A_P \#* P' \rangle \langle y \# P' \rangle \langle A_P \#* C \rangle P' \text{eq}$ 
ultimately have  $\text{Prop } C \Psi (\nu x)P ([x, y] \cdot p) \cdot (M(\nu*(\text{xvec}@y\#\text{yvec})) \langle ([x, y] \cdot N) \rangle) \langle ([x, y] \cdot p) \cdot [(x, y) \cdot P'] \rangle (x\#A_P) \Psi_P$ 
apply –
apply(rule rAlpha[where  $\alpha = M(\nu*(\text{xvec}@y\#\text{yvec})) \langle ([x, y] \cdot N) \rangle$ ])
apply(assumption | simp)+
apply(simp add: eqvts)
apply(assumption | simp add: abs-fresh)+
apply(simp add: fresh-left calc-atm)
apply(assumption | simp)+
apply(simp add: fresh-left calc-atm)
apply(assumption | simp)+
by(simp add: eqvts fresh-left)+
with  $\alpha \text{eq} P' \text{eq} D E F G H I$  show ?case
by(simp add: eqvts)
next
case(cBrOpen  $\Psi P M \text{xvec} \text{yvec} N P' x A_P \Psi_P C \alpha P''$ )
note  $\langle iM(\nu*(\text{xvec}@x\#\text{yvec})) \langle N \rangle \prec P' = \alpha \prec P'' \rangle$ 
moreover from  $\langle \text{xvec} \#* \alpha \rangle \langle x \# \alpha \rangle \langle \text{yvec} \#* \alpha \rangle$  have  $(\text{xvec}@x\#\text{yvec}) \#* (bn \alpha)$ 
by auto
moreover from  $\langle \text{xvec} \#* \text{yvec} \rangle \langle x \# \text{xvec} \rangle \langle x \# \text{yvec} \rangle \langle \text{distinct } \text{xvec} \rangle \langle \text{distinct } \text{yvec} \rangle$ 
have  $\text{distinct}(\text{xvec}@x\#\text{yvec})$ 
by(auto simp add: fresh-star-def) (simp add: fresh-def name-list-supp)
moreover note  $\langle \text{distinct}(bn \alpha) \rangle$ 
moreover from  $\langle \text{xvec} \#* M \rangle \langle x \# M \rangle \langle \text{yvec} \#* M \rangle$  have  $(\text{xvec}@x\#\text{yvec}) \#* M$ 
by auto
then have  $(\text{xvec}@x\#\text{yvec}) \#* (iM(\nu*(\text{xvec}@x\#\text{yvec})) \langle N \rangle \prec P')$  by auto
moreover from  $\langle bn \alpha \#* \text{subject } \alpha \rangle$  have  $bn \alpha \#* (\alpha \prec P'')$  by simp
ultimately obtain  $p$  where  $S: (\text{set } p) \subseteq (\text{set}(\text{xvec}@x\#\text{yvec})) \times (\text{set}(p \cdot (\text{xvec}@x\#\text{yvec})))$ 
and  $\text{distinctPerm } p$ 
and  $\alpha \text{eq}: \alpha = i(p \cdot M)(\nu*(p \cdot (\text{xvec}@x\#\text{yvec})) \langle (p \cdot N) \rangle)$  and  $P' \text{eq}: P'' = (p \cdot P')$ 
and  $A: (\text{xvec}@x\#\text{yvec}) \#* (i(p \cdot M)(\nu*(p \cdot (\text{xvec}@x\#\text{yvec})) \langle (p \cdot N) \rangle))$ 
and  $B: (p \cdot (\text{xvec}@x\#\text{yvec})) \#* (iM(\nu*(\text{xvec}@x\#\text{yvec})) \langle N \rangle)$ 
and  $C: (p \cdot (\text{xvec}@x\#\text{yvec})) \#* P'$ 
apply –

```



```

apply(rule residualEq)
by(assumption | simp)+

note  $\langle \Psi \triangleright P \mapsto_{\downarrow} M(\nu^*(xvec@yvec)) \rangle \langle N \rangle \prec P' \langle x \in (supp N) \rangle$ 

moreover {
  fix C
  from  $\langle xvec \#* M \rangle \langle yvec \#* M \rangle$  have  $(xvec@yvec) \#* M$  by simp
  moreover from  $\langle distinct\ xvec \rangle \langle distinct\ yvec \rangle \langle xvec \#* yvec \rangle$  have  $distinct(xvec@yvec)$ 
    by auto (simp add: fresh-star-def name-list-supp fresh-def)
  ultimately have Prop C  $\Psi P (\downarrow M(\nu^*(xvec@yvec)) \langle N \rangle) P' A_P \Psi_P$  using  $\langle A_P$ 
 $\#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$ 
    by(fastforce intro!: cBrOpen)
}

moreover obtain y::name where  $y \# \Psi$  and  $y \neq x$  and  $y \# P$  and  $y \# xvec$ 
and  $y \# yvec$  and  $y \# \alpha$  and  $y \# P'$  and  $y \# A_P$  and  $y \# \Psi_P$  and  $y \# M$  and  $y \#$ 
 $N$  and  $y \# C$  and  $y \# p$ 
  by(generate-fresh name) auto
  moreover note  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$ 
 $\langle xvec \#* M \rangle$ 
     $\langle yvec \#* \Psi \rangle \langle yvec \#* P \rangle \langle yvec \#* M \rangle \langle yvec \#* C \rangle \langle x \# C \rangle \langle xvec \#* C \rangle \langle distinct$ 
 $xvec \rangle \langle distinct\ yvec \rangle$ 
     $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle \langle distinct\ A_P \rangle \langle x \# A_P \rangle \langle xvec \#* yvec \rangle \langle xvec \#*$ 
 $\Psi_P \rangle$ 
     $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* yvec \rangle \langle A_P \#* N \rangle \langle A_P \#*$ 
 $P' \rangle \langle A_P \#* C \rangle$ 
  ultimately have Prop C  $\Psi ((\nu x)P) (\downarrow M(\nu^*(xvec@y\#yvec)) \langle [(x, y)] \cdot N \rangle)$ 
 $\langle [(x, y)] \cdot P' \rangle (x\#A_P) \Psi_P$ 
    by(metis rBrOpen)
  moreover have  $([(x, y)] \cdot p) \cdot [(x, y)] \cdot M = [(x, y)] \cdot p \cdot M$ 
    by(subst perm-compose[symmetric]) simp
  with  $\langle y \# M \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# M \rangle$  have D:  $([(x, y)] \cdot p) \cdot M = p \cdot M$ 
    by(auto simp add: eqvts freshChainSimps)
  moreover have  $([(x, y)] \cdot p) \cdot [(x, y)] \cdot xvec = [(x, y)] \cdot p \cdot xvec$ 
    by(subst perm-compose[symmetric]) simp
  with  $\langle y \# xvec \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# xvec \rangle$  have E:  $([(x, y)] \cdot p) \cdot xvec = p$ 
 $\cdot xvec$ 
    by(auto simp add: eqvts freshChainSimps)
  moreover have  $([(x, y)] \cdot p) \cdot [(x, y)] \cdot yvec = [(x, y)] \cdot p \cdot yvec$ 
    by(subst perm-compose[symmetric]) simp
  with  $\langle y \# yvec \rangle \langle x \# \alpha \rangle \alpha eq \langle y \# p \rangle \langle x \# yvec \rangle$  have F:  $([(x, y)] \cdot p) \cdot yvec = p$ 
 $\cdot yvec$ 
    by(auto simp add: eqvts freshChainSimps)
  moreover have  $([(x, y)] \cdot p) \cdot [(x, y)] \cdot x = [(x, y)] \cdot p \cdot x$ 
    by(subst perm-compose[symmetric]) simp
  with  $\langle y \neq x \rangle \langle y \# p \rangle$  have G:  $([(x, y)] \cdot p) \cdot y = p \cdot x$ 
    apply(simp add: freshChainSimps calc-atm)
    apply(subgoal-tac y  $\neq$  p  $\cdot$  x)
    apply(clarsimp)

```

```

using A  $\alpha$ eq
  apply(simp add: eqvts)
  apply(subst fresh-atm[symmetric])
  apply(simp only: freshChainSimps)
  by simp
moreover have  $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot N) = [(x, y)] \cdot p \cdot N$ 
  by(subst perm-compose[symmetric]) simp
with  $\langle y \# N \rangle \langle x \# \alpha \rangle \langle y \# p \rangle \alpha$ eq have H:  $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot N) = p \cdot N$ 
  by(auto simp add: eqvts freshChainSimps)
moreover have  $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot P') = [(x, y)] \cdot p \cdot P'$ 
  by(subst perm-compose[symmetric]) simp
with  $\langle y \# P' \rangle \langle x \# P'' \rangle \langle y \# p \rangle P'$ eq have I:  $(([(x, y)] \cdot p) \cdot [(x, y)] \cdot P') = p \cdot P'$ 
  by(auto simp add: eqvts freshChainSimps)
from  $\langle y \# p \rangle \langle y \neq x \rangle$  have  $y \neq p \cdot x$ 
  apply(subst fresh-atm[symmetric])
  apply(simp only: freshChainSimps)
  by simp
moreover from S have  $([(x, y)] \cdot \text{set } p) \subseteq [(x, y)] \cdot (\text{set}(xvec@x\#yvec) \times \text{set}(p \cdot (xvec@x\#yvec)))$ 
  by(simp)
with  $\langle y \neq p \cdot x \rangle \langle (((x, y)] \cdot p) \cdot y) = p \cdot x \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle \langle x \# yvec \rangle \langle y \# yvec \rangle \langle y \# p \rangle \langle x \# \alpha \rangle \alpha$ eq have
  set $([(x, y)] \cdot p) \subseteq \text{set}(xvec@y\#yvec) \times \text{set}(((x, y)] \cdot p) \cdot (xvec@y\#yvec)$ 
  by(simp add: eqvts calc-atm perm-compose)
moreover note  $\langle xvec \#* \Psi \rangle \langle yvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle yvec \#* P \rangle \langle xvec \#* M \rangle \langle yvec \#* M \rangle$ 
 $\langle yvec \#* C \rangle S \langle \text{distinctPerm } p \rangle \langle x \# C \rangle \langle xvec \#* C \rangle \langle xvec \#* \Psi_P \rangle \langle yvec \#* \Psi_P \rangle \langle x \# \Psi \rangle$ 
 $\langle A_P \#* xvec \rangle \langle x \# A_P \rangle \langle A_P \#* yvec \rangle \langle A_P \#* M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# M \rangle \langle x \# A_P \rangle \langle A_P \#* N \rangle$ 
 $A B C \alpha$ eq  $\langle A_P \#* \alpha \rangle \langle y \# \Psi \rangle \langle y \neq x \rangle \langle y \# P \rangle \langle y \# M \rangle \langle y \# \Psi_P \rangle \langle y \# C \rangle \langle xvec \#* \alpha \rangle \langle x \# \alpha \rangle \langle yvec \#* \alpha \rangle \langle y \# \alpha \rangle \langle A_P \#* P \rangle \langle A_P \#* \Psi \rangle \langle y \# A_P \rangle \langle y \# N \rangle \langle A_P \#* P' \rangle \langle y \# P' \rangle \langle A_P \#* C \rangle P'$ eq
ultimately have Prop C  $\Psi$  ( $\nu x$ )P  $(([(x, y)] \cdot p) \cdot (iM(\nu*(xvec@y\#yvec))\langle([(x, y)] \cdot N)\rangle))$ 
  apply -
  apply(rule rAlpha[where  $\alpha=iM(\nu*(xvec@y\#yvec))\langle([(x, y)] \cdot N)\rangle$ ])
    apply(assumption | simp)+
    apply(simp add: eqvts)
    apply(assumption | simp add: abs-fresh)+
    apply(simp add: fresh-left calc-atm)
    apply(assumption | simp)+
    apply(simp add: fresh-left calc-atm)
    apply(assumption | simp)+
  by(simp add: eqvts fresh-left)+
with  $\alpha$ eq P'eq D E F G H I show ?case
  by(simp add: eqvts)
next

```

case(*cScope* $\Psi P \alpha P' x A_P \Psi_P C \alpha' P''$)
note $\langle \alpha \prec ((\nu x)P') = \alpha' \prec P'' \rangle$
moreover from $\langle bn \alpha \#* \alpha' \rangle$ **have** $bn \alpha \#* (bn \alpha')$ **by** *auto*
moreover note $\langle distinct (bn \alpha) \rangle \langle distinct (bn \alpha') \rangle$
moreover from $\langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha' \#* subject \alpha' \rangle$
have $bn \alpha \#* (\alpha \prec ((\nu x)P'))$ **and** $bn \alpha' \#* (\alpha' \prec P'')$ **by** *simp+*
ultimately obtain p **where** $S: (set p) \subseteq (set (bn \alpha)) \times (set (bn(p \cdot \alpha)))$ **and**
distinctPerm p
and $\alpha Eq: \alpha' = p \cdot \alpha$ **and** $P' eq: P'' = p \cdot ((\nu x)P')$ **and** $(bn(p \cdot \alpha)) \#* \alpha$
and $(bn(p \cdot \alpha)) \#* ((\nu x)P')$
by(*rule residualEq*)

note $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle$
moreover from $\langle bn \alpha \#* subject \alpha \rangle \langle distinct (bn \alpha) \rangle$
have $\bigwedge C. Prop C \Psi P \alpha P' A_P \Psi_P$ **by**(*fastforce intro!: cScope*)

moreover note $\langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* \Psi_P \rangle$
 $\langle x \# C \rangle \langle bn \alpha \#* C \rangle \langle distinct (bn \alpha) \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle distinct A_P \rangle \langle x \# A_P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* P' \rangle \langle A_P \#* C \rangle$
ultimately have $Prop C \Psi ((\nu x)P) \alpha ((\nu x)P') (x \# A_P) \Psi_P$
by(*metis rScope*)

with $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle x \# \alpha \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* C \rangle \langle bn \alpha \#* (bn \alpha') \rangle$ $S \langle distinctPerm p \rangle \langle bn(p \cdot \alpha) \#* \alpha \rangle \langle bn(p \cdot \alpha) \#* ((\nu x)P') \rangle \langle A_P \#* \alpha \rangle$
 $\langle A_P \#* \alpha' \rangle \alpha Eq \langle x \# \alpha' \rangle \langle bn \alpha \#* \Psi_P \rangle \langle bn \alpha \#* \alpha' \rangle \langle x \# \Psi \rangle \langle A_P \#* \Psi \rangle \langle x \# A_P \rangle$
 $\langle A_P \#* P \rangle \langle A_P \#* P' \rangle \langle x \# C \rangle \langle A_P \#* C \rangle$
have $Prop C \Psi ((\nu x)P) (p \cdot \alpha) (p \cdot ((\nu x)P')) (x \# A_P) \Psi_P$
by(*fastforce intro!: rAlpha simp add: abs-fresh*)
with $\alpha Eq P' eq \langle distinctPerm p \rangle$ **show** *?case by simp*

next
case(*cBang* $\Psi P A_P \Psi_P C \alpha P'$)
then show *?case by*(*fastforce intro!: rBang*)

qed

lemma *inputFrameInduct*[*consumes 3, case-names cAlpha cInput cCase cPar1 cPar2 cScope cBang*]:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c) psi$
and $Prop :: 'f::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow 'a \Rightarrow 'a \Rightarrow ('a, 'b, 'c) psi \Rightarrow name list \Rightarrow 'b \Rightarrow bool$
and $C :: 'f::fs-name$

assumes $Trans: \Psi \triangleright P \mapsto M(N) \prec P'$
and $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$
and $distinct A_P$
and $rAlpha: \bigwedge \Psi P M N P' A_P \Psi_P p C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#*$

$N; A_P \#^* P'; A_P \#^* (p \cdot A_P); A_P \#^* C;$
 $\quad \text{set } p \subseteq \text{set } A_P \times \text{set}(p \cdot A_P); \text{distinctPerm } p;$
 $\quad \text{Prop } C \Psi P M N P' A_P \Psi_P \Longrightarrow \text{Prop } C \Psi$

$P M N P' (p \cdot A_P) (p \cdot \Psi_P)$
and $rInput: \bigwedge \Psi M K \text{ xvec } N \text{ Tvec } P C.$
 $\quad \llbracket \Psi \vdash M \leftrightarrow K; \text{distinct } \text{xvec}; \text{set } \text{xvec} \subseteq \text{supp } N;$
 $\quad \text{length } \text{xvec} = \text{length } \text{Tvec}; \text{xvec} \#^* \Psi;$
 $\quad \text{xvec} \#^* M; \text{xvec} \#^* K; \text{xvec} \#^* C \rrbracket \Longrightarrow$
 $\quad \text{Prop } C \Psi (M(\lambda * \text{xvec } N).P)$
 $\quad K (N[\text{xvec}::=\text{Tvec}]) (P[\text{xvec}::=\text{Tvec}]) (\Box) \mathbf{(1)}$

and $rCase: \bigwedge \Psi P M N P' \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto M(N) \prec P'; \text{extractFrame}$
 $P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P; \bigwedge C. \text{Prop } C \Psi P M N P' A_P \Psi_P;$
 $\quad (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P; \Psi_P \simeq \mathbf{1};$
 $(\text{supp } \Psi_P) = (\{\}::\text{name set});$

$A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* N; A_P$
 $\#^* P'; A_P \#^* C \rrbracket \Longrightarrow \text{Prop } C \Psi (\text{Cases } Cs) M N P' (\Box) \mathbf{(1)}$

and $rPar1: \bigwedge \Psi \Psi_Q P M N P' A_Q Q A_P \Psi_P C.$
 $\quad \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P';$
 $\quad \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\quad \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$
 $\quad \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_Q) P M N P' A_P \Psi_P;$
 $A_P \#^* P; A_P \#^* Q; A_P \#^* \Psi; A_P \#^* M; A_P \#^* N; A_P \#^* P'; A_P \#^*$
 $A_Q; A_P \#^* \Psi_Q;$
 $A_Q \#^* P; A_Q \#^* Q; A_Q \#^* \Psi; A_Q \#^* M; A_Q \#^* N; A_Q \#^* P'; A_Q$
 $\#^* \Psi_P;$

$A_P \#^* C; A_Q \#^* C \rrbracket \Longrightarrow$
 $\text{Prop } C \Psi (P \parallel Q) M N (P' \parallel Q) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and $rPar2: \bigwedge \Psi \Psi_P Q M N Q' A_P P A_Q \Psi_Q C.$
 $\quad \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(N) \prec Q';$
 $\quad \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\quad \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$
 $\quad \bigwedge C. \text{Prop } C (\Psi \otimes \Psi_P) Q M N Q' A_Q \Psi_Q;$
 $A_P \#^* P; A_P \#^* Q; A_P \#^* \Psi; A_P \#^* M; A_P \#^* N; A_P \#^* Q'; A_P$
 $\#^* A_Q; A_P \#^* \Psi_Q;$
 $A_Q \#^* P; A_Q \#^* Q; A_Q \#^* \Psi; A_Q \#^* M; A_Q \#^* N; A_Q \#^* Q'; A_Q$
 $\#^* \Psi_P;$

$A_P \#^* C; A_Q \#^* C \rrbracket \Longrightarrow$
 $\text{Prop } C \Psi (P \parallel Q) M N (P \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and $rScope: \bigwedge \Psi P M N P' x A_P \Psi_P C.$
 $\quad \llbracket \Psi \triangleright P \mapsto M(N) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$
 $\quad \bigwedge C. \text{Prop } C \Psi P M N P' A_P \Psi_P; x \# \Psi; x \# M; x \# N;$
 $\quad x \# A_P; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* N; A_P \#^* P';$
 $A_P \#^* C; x \# C \rrbracket \Longrightarrow$
 $\text{Prop } C \Psi ((\nu x)P) M N ((\nu x)P') (x \# A_P) \Psi_P$

and $rBang: \bigwedge \Psi P M N P' A_P \Psi_P C.$
 $\quad \llbracket \Psi \triangleright P \parallel !P \mapsto M(N) \prec P'; \text{guarded } P; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle; \text{distinct } A_P;$
 $\quad \bigwedge C. \text{Prop } C \Psi (P \parallel !P) M N P' A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; (\text{supp}$
 $\Psi_P) = (\{\}::\text{name set});$

```

       $A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* C \implies$ 
Prop C Ψ (!P) M N P' ([]) (1)
shows Prop C Ψ P M N P' A_P Ψ_P
  using Trans FrP ‹distinct A_P›
proof(nominal-induct Ψ P Rs==M(N) < P' A_P Ψ_P avoiding: C arbitrary: P'
rule: semanticsFrameInduct)
  case cAlpha
  then show ?case by (simp add: rAlpha)
next
  case cInput
  then show ?case by(auto simp add: rInput residualInject)
next
  case cBrInput
  then show ?case by(simp add: residualInject)
next
  case cOutput
  then show ?case by(simp add: residualInject)
next
  case cBrOutput
  then show ?case by(simp add: residualInject)
next
  case cCase
  then show ?case by(simp add: rCase residualInject)
next
  case cPar1
  then show ?case by(auto simp add: rPar1 residualInject)
next
  case cPar2
  then show ?case by(auto simp add: rPar2 residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(simp add: residualInject)
next
  case cOpen

```

```

then show ?case by(simp add: residualInject)
next
case cBrOpen
then show ?case by(simp add: residualInject)
next
case cScope
then show ?case by(auto simp add: rScope residualInject)
next
case cBang
then show ?case by(simp add: rBang residualInject)
qed

```

lemma *brinputFrameInduct*[*consumes 3, case-names cAlpha cBrInput cCase cPar1 cPar2 cBrMerge cScope cBang*]:

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $Prop$   :: 'f::fs-name  $\Rightarrow$  'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$ 
           'a  $\Rightarrow$  'a  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$  name list  $\Rightarrow$  'b  $\Rightarrow$  bool
and  $C$      :: 'f::fs-name

```

assumes *Trans*: $\Psi \triangleright P \mapsto_i M(N) \prec P'$

and *FrP*: $extractFrame P = \langle A_P, \Psi_P \rangle$

and *distinct* A_P

and *rAlpha*: $\bigwedge \Psi P M N P' A_P \Psi_P p C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* (p \cdot A_P); A_P \#* C \rrbracket \Longrightarrow$

$set p \subseteq set A_P \times set(p \cdot A_P); distinctPerm p;$
 $Prop C \Psi P M N P' A_P \Psi_P \rrbracket \Longrightarrow Prop C \Psi$

$P M N P' (p \cdot A_P) (p \cdot \Psi_P)$

and *rBrInput*: $\bigwedge \Psi M K xvec N Tvec P C.$

$\llbracket \Psi \vdash K \succeq M; distinct xvec; set xvec \subseteq supp N;$

$length xvec = length Tvec; xvec \#* \Psi;$

$xvec \#* M; xvec \#* K; xvec \#* C \rrbracket \Longrightarrow$

$Prop C \Psi (M(\lambda * xvec N).P)$

$K (N[xvec ::= Tvec]) (P[xvec ::= Tvec]) (\{\}) \mathbf{(1)}$

and *rCase*: $\bigwedge \Psi P M N P' \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto_i M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \bigwedge C. Prop C \Psi P M N P' A_P \Psi_P;$

$(\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P; \Psi_P \simeq \mathbf{1};$

$(supp \Psi_P) = (\{\}::name set);$

$A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* N; A_P$

$\#* P'; A_P \#* C \rrbracket \Longrightarrow Prop C \Psi (Cases Cs) M N P' (\{\}) \mathbf{(1)}$

and *rPar1*: $\bigwedge \Psi \Psi_Q P M N P' A_Q Q A_P \Psi_P C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P';$

$extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$

$extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$

$\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P M N P' A_P \Psi_P;$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#*$

$A_Q; A_P \#^* \Psi_Q;$
 $A_Q \#^* P; A_Q \#^* Q; A_Q \#^* \Psi; A_Q \#^* M; A_Q \#^* N; A_Q \#^* P'; A_Q$
 $\#^* \Psi_P;$
 $A_P \#^* C; A_Q \#^* C \implies$
 $Prop C \Psi (P \parallel Q) M N (P' \parallel Q) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rPar2: \bigwedge \Psi \Psi_P Q M N Q' A_P P A_Q \Psi_Q C.$
 $\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q' \rrbracket;$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q M N Q' A_Q \Psi_Q;$
 $A_P \#^* P; A_P \#^* Q; A_P \#^* \Psi; A_P \#^* M; A_P \#^* N; A_P \#^* Q'; A_P$
 $\#^* A_Q; A_P \#^* \Psi_Q;$
 $A_Q \#^* P; A_Q \#^* Q; A_Q \#^* \Psi; A_Q \#^* M; A_Q \#^* N; A_Q \#^* Q'; A_Q$
 $\#^* \Psi_P;$
 $A_P \#^* C; A_Q \#^* C \implies$
 $Prop C \Psi (P \parallel Q) M N (P \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rBrMerge: \bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'; \bigwedge C. Prop C (\Psi \otimes \Psi_Q) P M N$
 $P' A_P \Psi_P;$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'; \bigwedge C. Prop C (\Psi \otimes \Psi_P) Q M N$
 $Q' A_Q \Psi_Q;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_P \#^* \Psi; A_P \#^* \Psi_Q; A_P \#^* P; A_P \#^* N; A_P \#^* P';$
 $A_P \#^* Q; A_P \#^* Q'; A_P \#^* A_Q; A_P \#^* M; A_Q \#^* M;$
 $A_Q \#^* \Psi; A_Q \#^* \Psi_P; A_Q \#^* P; A_Q \#^* N; A_Q \#^* P';$
 $A_Q \#^* Q; A_Q \#^* Q'; A_P \#^* C; A_Q \#^* C \implies$
 $Prop C \Psi (P \parallel Q) M N (P' \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$
and $rScope: \bigwedge \Psi P M N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto \iota M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct$
 $A_P;$
 $\bigwedge C. Prop C \Psi P M N P' A_P \Psi_P; x \# \Psi; x \# M; x \# N;$
 $x \# A_P; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* N; A_P \#^* P';$
 $A_P \#^* C; x \# C \implies$
 $Prop C \Psi ((\nu x)P) M N ((\nu x)P') (x \# A_P) \Psi_P$
and $rBang: \bigwedge \Psi P M N P' A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto \iota M(N) \prec P'; guarded P; extractFrame P =$
 $\langle A_P, \Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi (P \parallel !P) M N P' A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; (supp$
 $\Psi_P) = (\{\}::name set);$
 $A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* N; A_P \#^* P'; A_P \#^* C \implies$
 $Prop C \Psi (!P) M N P' (\mathbf{1}) (\mathbf{1})$
shows $Prop C \Psi P M N P' A_P \Psi_P$
using $Trans FrP \langle distinct A_P \rangle$
proof(nominal-induct $\Psi P Rs == \iota M(N) \prec P' A_P \Psi_P$ avoiding: C arbitrary: P'
rule: semanticsFrameInduct)
case $cAlpha$
then show ?case by (simp add: $rAlpha$)
next

```

    case cInput
    then show ?case by(simp add: residualInject)
next
    case cBrInput
    then show ?case by(auto simp add: rBrInput residualInject)
next
    case cOutput
    then show ?case by(simp add: residualInject)
next
    case cBrOutput
    then show ?case by(simp add: residualInject)
next
    case cCase
    then show ?case by(simp add: rCase residualInject)
next
    case cPar1
    then show ?case by(auto simp add: rPar1 residualInject)
next
    case cPar2
    then show ?case by(auto simp add: rPar2 residualInject)
next
    case cComm1
    then show ?case by(simp add: residualInject)
next
    case cComm2
    then show ?case by(simp add: residualInject)
next
    case cBrMerge
    then show ?case by(auto simp add: rBrMerge residualInject)
next
    case cBrComm1
    then show ?case by(simp add: residualInject)
next
    case cBrComm2
    then show ?case by(simp add: residualInject)
next
    case cBrClose
    then show ?case by(simp add: residualInject)
next
    case cOpen
    then show ?case by(simp add: residualInject)
next
    case cBrOpen
    then show ?case by(simp add: residualInject)
next
    case cScope
    then show ?case by(auto simp add: rScope residualInject)
next
    case cBang

```


then show $?case$ **by** (*simp add: rBang residualInject*)
qed

lemma *outputFrameInduct*[*consumes 3, case-names cAlpha cOutput cCase cPar1 cPar2 cOpen cScope cBang*]:

fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c)$ *psi*
and M $:: 'a$
and B $:: ('a, 'b, 'c)$ *boundOutput*
and A_P $::$ *name list*
and Ψ_P $:: 'b$
and $Prop$ $:: 'f::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c)$ *psi* \Rightarrow
 $'a \Rightarrow ('a, 'b, 'c)$ *boundOutput* \Rightarrow *name list* $\Rightarrow 'b \Rightarrow bool$
and C $:: 'f::fs-name$

assumes *Trans*: $\Psi \triangleright P \mapsto ROut\ M\ B$

and *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$

and *distinct* A_P

and *rAlpha*: $\bigwedge \Psi\ P\ M\ A_P\ \Psi_P\ p\ B\ C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* (p \cdot A_P); A_P \#* B; A_P \#* C;$

$set\ p \subseteq set\ A_P \times set(p \cdot A_P); distinctPerm\ p;$
 $Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P \rrbracket \Longrightarrow Prop\ C\ \Psi\ P\ M\ B$

$(p \cdot A_P)\ (p \cdot \Psi_P)$

and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C. \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ C\ \Psi\ (M \langle N \rangle . P)\ K\ (N \prec' P)\ (\Box)\ (\mathbf{1})$

and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ A_P\ \Psi_P\ C. \llbracket \Psi \triangleright P \mapsto (ROut\ M\ B); extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P; \bigwedge C. Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P;$

$(\varphi, P) \in set\ Cs; \Psi \vdash \varphi; guarded\ P; \Psi_P \simeq \mathbf{1};$

$(supp\ \Psi_P) = (\{\}::name\ set);$

$A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* B; A_P \#* C \rrbracket \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B\ (\Box)\ (\mathbf{1})$

and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P';$

extractFrame $P = \langle A_P, \Psi_P \rangle; distinct\ A_P;$

extractFrame $Q = \langle A_Q, \Psi_Q \rangle; distinct\ A_Q;$

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ ((\nu * xvec)N \prec' P')\ A_P\ \Psi_P;$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* xvec; A_P \#* N; A_P \#* P'; A_P \#* A_Q; A_P \#* \Psi_Q;$

$A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* xvec; A_Q \#* N; A_Q \#* P'; A_Q \#* \Psi_P;$

$xvec \#* \Psi; xvec \#* P; xvec \#* Q; xvec \#* M; xvec \#* \Psi_P; xvec \#* \Psi_Q;$

$A_P \#* C; A_Q \#* C; xvec \#* C \rrbracket \Longrightarrow$

$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\nu * xvec)N \prec' (P' \parallel Q))\ (A_P @ A_Q)\ (\Psi_P \otimes$

$\Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ xvec\ N\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$

$\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec Q';$

extractFrame $P = \langle A_P, \Psi_P \rangle; distinct\ A_P;$

extractFrame $Q = \langle A_Q, \Psi_Q \rangle; distinct\ A_Q;$

$\bigwedge C. Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\nu * xvec)N \prec' Q')\ A_Q\ \Psi_Q;$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* xvec; A_P \#* N; A_P$
 $\#* Q'; A_P \#* A_Q; A_P \#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* xvec; A_Q \#* N; A_Q$
 $\#* Q'; A_Q \#* \Psi_P;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* Q; xvec \#* M; xvec \#* \Psi_P; xvec \#* \Psi_Q;$
 $A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu*xvec)N \prec' (P \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes$
 $\Psi_Q)$
and *rOpen*: $\bigwedge \Psi P M xvec yvec N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto M((\nu*(xvec @ yvec))\langle N \rangle \prec P'); extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi P M ((\nu*(xvec @ yvec))\langle N \rangle \prec P') A_P \Psi_P; x \in supp$
 $N; x \# \Psi; x \# M;$
 $x \# A_P; x \# xvec; x \# yvec; A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#*$
 $N; A_P \#* P';$
 $A_P \#* xvec; A_P \#* yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* \Psi_P;$
 $yvec \#* \Psi; yvec \#* P; yvec \#* M; A_P \#* C; x \# C; xvec \#* C; yvec$
 $\#* C \implies$
 $Prop C \Psi ((\nu x)P) M ((\nu*(xvec @ x\#yvec))\langle N \rangle \prec P') (x\#A_P) \Psi_P$
and *rScope*: $\bigwedge \Psi P M xvec N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto M((\nu*xvec)\langle N \rangle \prec P'); extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\bigwedge C. Prop C \Psi P M ((\nu*xvec)\langle N \rangle \prec P') A_P \Psi_P;$
 $x \# \Psi; x \# M; x \# xvec; x \# N; x \# A_P; A_P \#* \Psi; A_P \#* P;$
 $A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* xvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* \Psi_P;$
 $A_P \#* C; x \# C; xvec \#* C \implies$
 $Prop C \Psi ((\nu x)P) M ((\nu*xvec)\langle N \rangle \prec' ((\nu x)P')) (x\#A_P) \Psi_P$
and *rBang*: $\bigwedge \Psi P M B A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto ROut M B; guarded P; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi (P \parallel !P) M B A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; supp \Psi_P$
 $= (\{\}::name set);$
 $A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* C \implies Prop C \Psi (!P) M$
 $B (\parallel) (\mathbf{1})$
shows $Prop C \Psi P M B A_P \Psi_P$
proof –
 $\{$
 $\quad \mathbf{fix} B$
 $\quad \mathbf{assume} \Psi \triangleright P \mapsto ROut M B$
 $\quad \mathbf{then\ have} Prop C \Psi P M B A_P \Psi_P \mathbf{using} FrP \langle distinct A_P \rangle$
 $\quad \mathbf{proof} (nominal-induct \Psi P Rs == ROut M B A_P \Psi_P \mathbf{avoiding:} C \mathbf{arbitrary:} B$
rule: semanticsFrameInduct)
 $\quad \mathbf{case} cAlpha$
 $\quad \mathbf{then\ show} ?case \mathbf{by} (auto \mathbf{intro:} rAlpha)$
 $\quad \mathbf{next}$
 $\quad \mathbf{case} cInput$
 $\quad \mathbf{then\ show} ?case \mathbf{by} (simp \mathbf{add:} residualInject)$

```

next
  case cBrInput
  then show ?case by(simp add: residualInject)
next
  case cOutput
  then show ?case by(force intro: rOutput simp add: residualInject)
next
  case cBrOutput
  then show ?case by(simp add: residualInject)
next
  case cCase
  then show ?case by(force intro: rCase simp add: residualInject)
next
  case cPar1
  then show ?case
    by(auto intro!: rPar1 simp add: residualInject)
next
  case cPar2
  then show ?case
    by(auto intro!: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(simp add: residualInject)
next
  case cOpen
  then show ?case by(auto intro: rOpen simp add: residualInject)
next
  case cBrOpen
  then show ?case by(simp add: residualInject)
next
  case cScope
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case cBang

```

then show $?case$ **by**($force$ $intro$: $rBang$ $simp$ add : $residualInject$)
qed
with $Trans$ **show** $?thesis$ **by**($simp$ add : $residualInject$)
qed

lemma $broutputFrameInduct$ [$consumes$ 3, $case$ -names $cAlpha$ $cBrOutput$ $cCase$ $cPar1$ $cPar2$ $cBrComm1$ $cBrComm2$ $cBrOpen$ $cScope$ $cBang$]:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and B :: ('a, 'b, 'c) $boundOutput$
and A_P :: $name$ $list$
and Ψ_P :: 'b
and $Prop$:: 'f:: fs -name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow
' a \Rightarrow ('a, 'b, 'c) $boundOutput$ \Rightarrow $name$ $list$ \Rightarrow 'b \Rightarrow $bool$
and C :: 'f:: fs -name

assumes $Trans$: $\Psi \triangleright P \mapsto RBrOut M B$

and FrP : $extractFrame P = \langle A_P, \Psi_P \rangle$

and $distinct A_P$

and $rAlpha$: $\bigwedge \Psi P M A_P \Psi_P p B C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* (p \cdot A_P); A_P \#* B; A_P \#* C;$

$set p \subseteq set A_P \times set(p \cdot A_P); distinctPerm p;$
 $Prop C \Psi P M B A_P \Psi_P \rrbracket \Longrightarrow Prop C \Psi P M B$

$(p \cdot A_P) (p \cdot \Psi_P)$

and $rBrOutput$: $\bigwedge \Psi M K N P C. \Psi \vdash M \preceq K \Longrightarrow Prop C \Psi (M \langle N \rangle . P) K$
 $(N \prec' P) (\llbracket \rrbracket) (1)$

and $rCase$: $\bigwedge \Psi P M B \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto (RBrOut M B); extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \bigwedge C. Prop C \Psi P M B A_P \Psi_P;$

$(\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P; \Psi_P \simeq \mathbf{1};$

$(supp \Psi_P) = (\{ \} :: name set);$

$A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* B; A_P \#* C \rrbracket \Longrightarrow Prop C \Psi (Cases Cs) M B (\llbracket \rrbracket) (1)$

and $rPar1$: $\bigwedge \Psi \Psi_Q P M xvec N P' A_Q Q A_P \Psi_P C.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P';$

$extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$

$extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$

$\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P M ((\nu * xvec) N \prec' P') A_P \Psi_P;$

$A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* xvec; A_P \#* N; A_P \#* P'; A_P \#* A_Q; A_P \#* \Psi_Q;$

$A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* xvec; A_Q \#* N; A_Q \#* P'; A_Q \#* \Psi_P;$

$xvec \#* \Psi; xvec \#* P; xvec \#* Q; xvec \#* M; xvec \#* \Psi_P; xvec \#* \Psi_Q;$

$A_P \#* C; A_Q \#* C; xvec \#* C \rrbracket \Longrightarrow$

$Prop C \Psi (P \parallel Q) M ((\nu * xvec) N \prec' (P' \parallel Q)) (A_P @ A_Q) (\Psi_P \otimes$

$\Psi_Q)$

and $rPar2$: $\bigwedge \Psi \Psi_P Q M xvec N Q' A_P P A_Q \Psi_Q C.$

$\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q';$

$extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\wedge C. Prop C (\Psi \otimes \Psi_P) Q M ((\nu * xvec)N \prec' Q') A_Q \Psi_Q;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* M; A_P \#* xvec; A_P \#* N; A_P$
 $\#* Q'; A_P \#* A_Q; A_P \#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* M; A_Q \#* xvec; A_Q \#* N; A_Q$
 $\#* Q'; A_Q \#* \Psi_P;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* Q; xvec \#* M; xvec \#* \Psi_P; xvec \#* \Psi_Q;$
 $A_P \#* C; A_Q \#* C; xvec \#* C \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu * xvec)N \prec' (P \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes$
 $\Psi_Q)$
and $rBrComm1: \wedge \Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M((\nu * xvec)N) \prec Q'; extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle; distinct A_Q;$
 $distinct xvec;$
 $\wedge C. Prop C (\Psi \otimes \Psi_P) Q M ((\nu * xvec)N \prec' Q') A_Q \Psi_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu * xvec)N \prec' (P' \parallel Q')) (A_P @ A_Q) (\Psi_P$
 $\otimes \Psi_Q)$
and $rBrComm2: \wedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M((\nu * xvec)N) \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\wedge C. Prop C (\Psi \otimes \Psi_Q) P M ((\nu * xvec)N \prec' P') A_P \Psi_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P;$
 $xvec \#* Q; A_P \#* C; A_Q \#* C; xvec \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M \implies$
 $Prop C \Psi (P \parallel Q) M ((\nu * xvec)N \prec' (P' \parallel Q')) (A_P @ A_Q) (\Psi_P$
 $\otimes \Psi_Q)$
and $rBrOpen: \wedge \Psi P M xvec yvec N P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto_i M((\nu * (xvec @ yvec))N) \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\wedge C. Prop C \Psi P M ((\nu * (xvec @ yvec))N \prec' P') A_P \Psi_P; x \in supp$
 $N; x \# \Psi; x \# M;$
 $x \# A_P; x \# xvec; x \# yvec; A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#*$
 $N; A_P \#* P';$
 $A_P \#* xvec; A_P \#* yvec;$

```

      xvec #* Ψ; xvec #* P; xvec #* M; xvec #* ΨP;
      yvec #* Ψ; yvec #* P; yvec #* M; AP #* C; x # C; xvec #* C; yvec
#* C]] ⇒
      Prop C Ψ ((νx)P) M ((ν*(xvec@x#yvec))N <' P') (x#AP) ΨP
and rScope: ∧Ψ P M xvec N P' x AP ΨP C.
      [[Ψ ▷ P ↦iM(ν*xvec)⟨N⟩ < P'; extractFrame P = ⟨AP, ΨP⟩;
distinct AP;
      ∧C. Prop C Ψ P M ((ν*xvec)N <' P') AP ΨP;
      x # Ψ; x # M; x # xvec; x # N; x # AP; AP #* Ψ; AP #* P;
      AP #* M; AP #* N; AP #* P'; AP #* xvec;
      xvec #* Ψ; xvec #* P; xvec #* M; xvec #* ΨP;
      AP #* C; x # C; xvec #* C]] ⇒
      Prop C Ψ ((νx)P) M ((ν*xvec)N <' ((νx)P')) (x#AP) ΨP
and rBang: ∧Ψ P M B AP ΨP C.
      [[Ψ ▷ P || !P ↦iRBrOut M B; guarded P; extractFrame P =
⟨AP, ΨP⟩; distinct AP;
      ∧C. Prop C Ψ (P || !P) M B AP (ΨP ⊗ 1); ΨP ≃ 1; supp ΨP
= ({}::name set);
      AP #* Ψ; AP #* P; AP #* M; AP #* C]] ⇒ Prop C Ψ (!P) M
B ([]) (1)
shows Prop C Ψ P M B AP ΨP
proof –
  {
    fix B
    assume Ψ ▷ P ↦iRBrOut M B
    then have Prop C Ψ P M B AP ΨP using FrP <distinct AP>
    proof(nominal-induct Ψ P Rs==RBrOut M B AP ΨP avoiding: C arbitrary:
B rule: semanticsFrameInduct)
      case cAlpha
      then show ?case by(auto intro: rAlpha)
    next
      case cInput
      then show ?case by(simp add: residualInject)
    next
      case cBrInput
      then show ?case by(simp add: residualInject)
    next
      case cOutput
      then show ?case by(simp add: residualInject)
    next
      case cBrOutput
      then show ?case by(force intro: rBrOutput simp add: residualInject)
    next
      case cCase
      then show ?case by(force intro: rCase simp add: residualInject)
    next
      case cPar1
      then show ?case by(auto intro!: rPar1 simp add: residualInject)
    next

```

```

    case cPar2
  then show ?case by(auto intro!: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(simp add: residualInject)
next
  case cComm2
  then show ?case by(simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(auto intro: rBrComm1 simp add: residualInject)
next
  case cBrComm2
  then show ?case by(auto intro: rBrComm2 simp add: residualInject)
next
  case cBrClose
  then show ?case by(simp add: residualInject)
next
  case cOpen
  then show ?case by(simp add: residualInject)
next
  case cBrOpen
  then show ?case by(auto intro: rBrOpen simp add: residualInject)
next
  case cScope
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case cBang
  then show ?case by(force intro: rBang simp add: residualInject)
qed
}
with Trans show ?thesis by(simp add: residualInject)
qed

```

lemma *tauFrameInduct*[*consumes 3, case-names cAlpha cCase cPar1 cPar2 cComm1 cComm2 cBrClose cScope cBang*]:

```

fixes  $\Psi$     :: 'b
and  $P$       :: ('a, 'b, 'c) psi
and  $P'$      :: ('a, 'b, 'c) psi
and  $Prop$  :: 'f::fs-name  $\Rightarrow$  'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$ 
           ('a, 'b, 'c) psi  $\Rightarrow$  name list  $\Rightarrow$  'b  $\Rightarrow$  bool
and  $C$       :: 'f::fs-name

```

```

assumes Trans:  $\Psi \triangleright P \mapsto_{\tau} \prec P'$ 
and  $FrP$ : extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
and distinct  $A_P$ 

```

and *rAlpha*: $\bigwedge \Psi P P' A_P \Psi_P p C. \llbracket A_P \#* \Psi; A_P \#* P; A_P \#* P'; A_P \#* (p \cdot A_P); A_P \#* C; \rrbracket$
 $set p \subseteq set A_P \times set (p \cdot A_P); distinctPerm p;$
 $Prop C \Psi P P' A_P \Psi_P \rrbracket \implies Prop C \Psi P P' (p \cdot A_P) (p \cdot \Psi_P)$

and *rCase*: $\bigwedge \Psi P P' \varphi Cs A_P \Psi_P C. \llbracket \Psi \triangleright P \mapsto \tau \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \bigwedge C. Prop C \Psi P P' A_P \Psi_P; \rrbracket$
 $(\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P; \Psi_P \simeq \mathbf{1};$
 $(supp \Psi_P) = (\{\}::name set);$
 $A_P \#* \Psi; A_P \#* P; A_P \#* P'; A_P \#* C \rrbracket \implies$
 $Prop C \Psi (Cases Cs) P' (\llbracket \rrbracket) (\mathbf{1})$

and *rPar1*: $\bigwedge \Psi \Psi_Q P P' A_Q Q A_P \Psi_P C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \tau \prec P'; \rrbracket$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_Q) P P' A_P \Psi_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* P'; A_P \#* A_Q; A_P \#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* P'; A_Q \#* \Psi_P;$
 $A_P \#* C; A_Q \#* C \rrbracket \implies$
 $Prop C \Psi (P \parallel Q) (P' \parallel Q) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi \Psi_P Q Q' A_P P A_Q \Psi_Q C.$
 $\llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \tau \prec Q'; \rrbracket$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\bigwedge C. Prop C (\Psi \otimes \Psi_P) Q Q' A_Q \Psi_Q;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* Q'; A_P \#* A_Q; A_P \#* \Psi_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* Q'; A_Q \#* \Psi_P;$
 $A_P \#* C; A_Q \#* C \rrbracket \implies$
 $Prop C \Psi (P \parallel Q) (P \parallel Q') (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and *rComm1*: $\bigwedge \Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; \rrbracket$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec) \langle N \rangle \prec Q'; extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle; distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#*$
 $M;$
 $xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C \rrbracket \implies$
 $Prop C \Psi (P \parallel Q) ((\nu * xvec)(P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and *rComm2*: $\bigwedge \Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q C.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'; extractFrame P = \langle A_P,$
 $\Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P';$

$A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi; A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* K; A_Q \#* Q';$
 $A_Q \#* xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#*$

$M;$

$xvec \#* Q; xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C] \implies$
 $Prop C \Psi (P \parallel Q) ((\nu xvec)(P' \parallel Q')) (A_P @ A_Q) (\Psi_P \otimes \Psi_Q)$

and $rBrClose: \bigwedge \Psi P M xvec N P' A_P \Psi_P x C.$
 $\llbracket \Psi \triangleright P \mapsto \downarrow M(\nu xvec) \langle N \rangle \prec P';$
 $x \in supp M;$
 $extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* xvec;$
 $distinct xvec; xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P;$
 $xvec \#* M;$
 $x \# \Psi; x \# xvec; x \# A_P;$
 $A_P \#* C; xvec \#* C; x \# C] \implies$
 $Prop C \Psi ((\nu x)P) ((\nu x)(\nu xvec)P') (x \# A_P) \Psi_P$

and $rScope: \bigwedge \Psi P P' x A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \mapsto \tau \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\bigwedge C. Prop C \Psi P P' A_P \Psi_P; x \# \Psi;$
 $x \# A_P; A_P \#* \Psi; A_P \#* P; A_P \#* P';$
 $A_P \#* C; x \# C] \implies$
 $Prop C \Psi ((\nu x)P) ((\nu x)P') (x \# A_P) \Psi_P$

and $rBang: \bigwedge \Psi P P' A_P \Psi_P C.$
 $\llbracket \Psi \triangleright P \parallel !P \mapsto \tau \prec P'; guarded P; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\bigwedge C. Prop C \Psi (P \parallel !P) P' A_P (\Psi_P \otimes \mathbf{1}); \Psi_P \simeq \mathbf{1}; supp \Psi_P =$
 $(\{\} :: name set);$
 $A_P \#* \Psi; A_P \#* P; A_P \#* P'; A_P \#* C] \implies Prop C \Psi (!P) P'$

$(\square) (1)$

shows $Prop C \Psi P P' A_P \Psi_P$
using $Trans FrP \langle distinct A_P \rangle$

proof(*nominal-induct* $\Psi P Rs == \tau \prec P' A_P \Psi_P$ *avoiding: C arbitrary: P' rule:*
semanticsFrameInduct)

case $cAlpha$
then show $?case \text{ by}(force \text{ intro: } rAlpha \text{ simp add: residualInject})$
next

case $cInput$
then show $?case \text{ by}(simp \text{ add: residualInject})$
next

case $cBrInput$
then show $?case \text{ by}(simp \text{ add: residualInject})$
next

case $cOutput$
then show $?case \text{ by}(simp \text{ add: residualInject})$
next

case $cBrOutput$
then show $?case \text{ by}(simp \text{ add: residualInject})$
next

case $cCase$

```

  then show ?case by(force intro: rCase simp add: residualInject)
next
  case cPar1
  then show ?case by(force intro: rPar1 simp add: residualInject)
next
  case cPar2
  then show ?case by(force intro: rPar2 simp add: residualInject)
next
  case cComm1
  then show ?case by(force intro: rComm1 simp add: residualInject)
next
  case cComm2
  then show ?case by(force intro: rComm2 simp add: residualInject)
next
  case cBrMerge
  then show ?case by(simp add: residualInject)
next
  case cBrComm1
  then show ?case by(simp add: residualInject)
next
  case cBrComm2
  then show ?case by(simp add: residualInject)
next
  case cBrClose
  then show ?case by(force intro: rBrClose simp add: residualInject)
next
  case cOpen
  then show ?case by(simp add: residualInject)
next
  case cBrOpen
  then show ?case by(simp add: residualInject)
next
  case cScope
  then show ?case by(force intro: rScope simp add: residualInject)
next
  case cBang
  then show ?case by(force intro: rBang simp add: residualInject)
qed

```

lemma *inputFreshDerivative*:

```

fixes  $\Psi$  :: 'b
  and  $P$  :: ('a, 'b, 'c) psi
  and  $M$  :: 'a
  and  $N$  :: 'a
  and  $P'$  :: ('a, 'b, 'c) psi
  and  $x$  :: name

```

```

assumes  $\Psi \triangleright P \mapsto M(N) \prec P'$ 
  and  $x \# P$ 

```

```

and x # N

shows x # P'
proof -
  have  $bn(M(N)) \#* subject(M(N))$  and  $distinct(bn(M(N)))$  by simp+
  with  $\langle \Psi \triangleright P \mapsto M(N) \prec P' \rangle$  show ?thesis using  $\langle x \# P \rangle \langle x \# N \rangle$ 
  proof (nominal-induct  $\Psi P \alpha == M(N) P'$  avoiding: x rule: semanticsInduct)
    case (cAlpha  $\Psi P \alpha P' p x$ )
      then show ?case by simp
    next
      case (cInput  $\Psi M' K xvec N' Tvec P x$ )
        from  $\langle K(N[xvec::=Tvec]) = M(N) \rangle$  have  $M = K$  and  $NeqN': N =$ 
 $N[xvec::=Tvec]$  by (simp add: action.inject)+
        note  $\langle length\ xvec = length\ Tvec \rangle \langle distinct\ xvec \rangle$  then
        moreover have  $x \# Tvec$  using  $\langle set\ xvec \subseteq supp\ N' \rangle \langle x \# N \rangle NeqN'$ 
        by (blast intro: substTerm.subst3)
        moreover from  $\langle xvec \#* x \rangle \langle x \# M'(\lambda*xvec\ N').P \rangle$ 
        have  $x \# P$  by (simp add: inputChainFresh) (simp add: name-list-supp fresh-def)
        ultimately show ?case using  $\langle xvec \#* x \rangle$  by auto
      next
        case cBrInput
        then show ?case by simp
      next
        case (cOutput  $\Psi M K N P x$ )
        then show ?case by simp
      next
        case cBrOutput
        then show ?case by simp
      next
        case (cCase  $\Psi P P' \varphi Cs x$ )
        then show ?case by (induct Cs, auto)
      next
        case (cPar1  $\Psi \Psi_Q P P' xvec Q x$ )
        then show ?case by simp
      next
        case (cPar2  $\Psi \Psi_P Q Q' xvec P x$ )
        then show ?case by simp
      next
        case (cComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q x$ )
        then show ?case by simp
      next
        case (cComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q x$ )
        then show ?case by simp
      next
        case cBrMerge
        then show ?case by simp
      next
        case cBrComm1
        then show ?case by simp

```

```

next
  case cBrComm2
  then show ?case by simp
next
  case cBrClose
  then show ?case by simp
next
  case(cOpen  $\Psi$  P M xvec yvec N P' x y)
  then show ?case by simp
next
  case(cBrOpen  $\Psi$  P M xvec yvec N P' x y)
  then show ?case by simp
next
  case(cScope  $\Psi$  P P' x y)
  then show ?case by(simp add: abs-fresh)
next
  case(cBang  $\Psi$  P P' x)
  then show ?case by simp
qed
qed

```

lemma *brinputFreshDerivative*:

```

fixes  $\Psi$  :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and x :: name

```

```

assumes  $\Psi \triangleright P \mapsto \iota M(\downarrow N) \prec P'$ 
and  $x \# P$ 
and  $x \# N$ 

```

shows $x \# P'$

proof –

```

have  $bn(\iota M(\downarrow N)) \#^* \text{subject}(\iota M(\downarrow N))$  and  $\text{distinct}(bn(\iota M(\downarrow N)))$  by simp+
with  $\langle \Psi \triangleright P \mapsto \iota M(\downarrow N) \prec P' \rangle$  show ?thesis using  $\langle x \# P \rangle \langle x \# N \rangle$ 
proof(nominal-induct  $\Psi$  P  $\alpha = \iota M(\downarrow N)$  P' avoiding: x rule: semanticsInduct)
  case(cAlpha  $\Psi$  P  $\alpha$  P' p x)
  then show ?case by simp
next
  case(cInput  $\Psi$  M' K xvec N' Tvec P x)
  then show ?case by simp
next
  case(cBrInput  $\Psi$  M' K xvec N' Tvec P x)
  from  $\langle \iota M'(\downarrow(N'[xvec ::= Tvec])) = \iota M(\downarrow N) \rangle$  have  $M' = M$  and  $\text{NeqN}' : N =$ 
 $N'[xvec ::= Tvec]$  by(simp add: action.inject)+
  note  $\langle \text{length } xvec = \text{length } Tvec \rangle \langle \text{distinct } xvec \rangle$  then
  moreover have  $x \# Tvec$  using  $\langle \text{set } xvec \subseteq \text{supp } N' \rangle \langle x \# N \rangle \text{NeqN}'$ 

```

```

    by(blast intro: substTerm.subst3)
    moreover from  $\langle xvec \#* x \rangle \langle x \# K(\lambda*xvec N') . P \rangle$ 
    have  $x \# P$  by(simp add: inputChainFresh) (simp add: name-list-supp fresh-def)
    ultimately show  $?case$  using  $\langle xvec \#* x \rangle$  by auto
next
  case(cOutput  $\Psi M K N P x$ )
  then show  $?case$  by simp
next
  case cBrOutput
  then show  $?case$  by simp
next
  case(cCase  $\Psi P P' \varphi Cs x$ )
  then show  $?case$  by(induct Cs, auto)
next
  case(cPar1  $\Psi \Psi_Q P P' xvec Q x$ )
  then show  $?case$  by simp
next
  case(cPar2  $\Psi \Psi_P Q Q' xvec P x$ )
  then show  $?case$  by simp
next
  case(cComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q x$ )
  then show  $?case$  by simp
next
  case(cComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q x$ )
  then show  $?case$  by simp
next
  case cBrMerge
  then show  $?case$  by simp
next
  case cBrComm1
  then show  $?case$  by simp
next
  case cBrComm2
  then show  $?case$  by simp
next
  case cBrClose
  then show  $?case$  by simp
next
  case(cOpen  $\Psi P M xvec yvec N P' x y$ )
  then show  $?case$  by simp
next
  case(cBrOpen  $\Psi P M xvec yvec N P' x y$ )
  then show  $?case$  by simp
next
  case(cScope  $\Psi P P' x y$ )
  then show  $?case$  by(simp add: abs-fresh)
next
  case(cBang  $\Psi P P' x$ )
  then show  $?case$  by simp

```

qed
qed

lemma *inputFreshChainDerivative*:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $xvec$:: name list

assumes $\Psi \triangleright P \mapsto M(N) \prec P'$
and $xvec \#* P$
and $xvec \#* N$

shows $xvec \#* P'$
using *assms*
by(*induct xvec*)
(*auto intro: inputFreshDerivative*)

lemma *brinputFreshChainDerivative*:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $xvec$:: name list

assumes $\Psi \triangleright P \mapsto_{\iota} M(N) \prec P'$
and $xvec \#* P$
and $xvec \#* N$

shows $xvec \#* P'$
using *assms*
by(*induct xvec*)
(*auto intro: brinputFreshDerivative*)

lemma *outputFreshDerivativeN*:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and $xvec$:: name list
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and x :: name

assumes $\Psi \triangleright P \mapsto M(\nu * xvec)(N) \prec P'$
and $xvec \#* M$
and *distinct xvec*

```

and  $x \# P$ 
and  $x \# xvec$ 

shows  $x \# N$ 
proof –
  note  $\langle \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P' \rangle$ 
  moreover from  $\langle xvec \#* M \rangle$  have  $bn(M(\nu*xvec)\langle N \rangle) \#* subject(M(\nu*xvec)\langle N \rangle)$ 
by simp
  moreover from  $\langle distinct\ xvec \rangle$  have  $distinct(bn(M(\nu*xvec)\langle N \rangle))$  by simp
  ultimately show  $freshN: x \# N$  using  $\langle x \# P \rangle \langle x \# xvec \rangle$ 
  proof(nominal-induct  $\Psi P \alpha == M(\nu*xvec)\langle N \rangle P'$  avoiding: x arbitrary: M xvec
N rule: semanticsInduct)
    case(cAlpha  $\Psi P \alpha P' p x M xvec N$ )
      have  $S: set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$  by fact
      from  $\langle (p \cdot \alpha) = M(\nu*xvec)\langle N \rangle \rangle$  have  $\langle (p \cdot p \cdot \alpha) = p \cdot (M(\nu*xvec)\langle N \rangle) \rangle$ 
by(simp add: fresh-star-bij)
      with  $\langle distinctPerm\ p \rangle$  have  $\alpha = (p \cdot M)(\nu*(p \cdot xvec))\langle (p \cdot N) \rangle$  by simp
      moreover from  $\langle (p \cdot \alpha) = M(\nu*xvec)\langle N \rangle \rangle \langle x \# xvec \rangle$  have  $x \# (bn(p \cdot \alpha))$  by
simp
      with  $\langle (bn\ \alpha) \#* x \rangle \langle x \# xvec \rangle S$  have  $x \# (p \cdot xvec)$ 
      by(fastforce dest: pt-fresh-bij1[OF pt-name-inst, OF at-name-inst, where
pi=p and x=xvec])
      ultimately have  $x \# (p \cdot N)$  using  $\langle x \# P \rangle$  by(metis cAlpha)
      then have  $(p \cdot x) \# (p \cdot p \cdot N)$  by(simp add: pt-fresh-bij1[OF pt-name-inst,
OF at-name-inst])
      with  $\langle distinctPerm\ p \rangle \langle bn(\alpha) \#* x \rangle \langle x \# (bn(p \cdot \alpha)) \rangle S$  show  $?case$  by simp
    next
      case cInput
      then show  $?case$  by simp
    next
      case cBrInput
      then show  $?case$  by simp
    next
      case cOutput
      then show  $?case$  by(simp add: action.inject)
    next
      case cBrOutput
      then show  $?case$  by(simp add: action.inject)
    next
      case (cCase  $\Psi P P' \varphi Cs x M xvec N$ )
      then show  $?case$  by(auto simp add: action.inject dest: memFresh)
    next
      case cPar1
      then show  $?case$  by simp
    next
      case cPar2
      then show  $?case$  by simp
    next
      case cComm1

```

```

    then show ?case by simp
  next
    case cComm2
    then show ?case by simp
  next
    case cBrMerge
    then show ?case by simp
  next
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose
    then show ?case by simp
  next
    case (cOpen  $\Psi$   $P$   $M$   $xvec$   $yvec$   $N$   $P'$   $x$   $y$   $M'$   $zvec$   $N'$ )
    from  $\langle M(\nu^*(xvec@x\#yvec))\langle N \rangle = M'(\nu^*zvec)\langle N' \rangle \rangle$  have  $zvec = xvec@x\#yvec$ 
  and  $N = N'$ 
    by (simp add: action.inject)+
    from  $\langle y \# (\nu x)P \rangle \langle x \# y \rangle$  have  $y \# P$  by (simp add: abs-fresh)
    moreover from  $\langle y \# zvec \rangle \langle zvec = xvec@x\#yvec \rangle$  have  $y \# (xvec@yvec)$ 
    by simp
    ultimately have  $y \# N$  by (fastforce intro!: cOpen)
    with  $\langle N = N' \rangle$  show ?case by simp
  next
    case cBrOpen
    then show ?case by simp
  next
    case cScope
    then show ?case by (auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
qed
qed

```

lemma *broutputFreshDerivativeN*:

```

  fixes  $\Psi$     :: 'b
  and  $P$       :: ('a, 'b, 'c) psi
  and  $M$      :: 'a
  and  $xvec$   :: name list
  and  $N$      :: 'a
  and  $P'$    :: ('a, 'b, 'c) psi
  and  $x$      :: name

```

```

assumes  $\Psi \triangleright P \mapsto_i M(\nu^*xvec)\langle N \rangle \prec P'$ 
and  $xvec \#^* M$ 

```



```

and distinct xvec
and x # P
and x # xvec

shows x # N
proof -
  note ⟨Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P'⟩
  moreover from ⟨xvec #* M⟩ have bn(iM(ν*xvec)⟨N⟩) #* subject(iM(ν*xvec)⟨N⟩)
  by simp
  moreover from ⟨distinct xvec⟩ have distinct(bn(iM(ν*xvec)⟨N⟩)) by simp
  ultimately show freshN: x # N using ⟨x # P⟩ ⟨x # xvec⟩
  proof (nominal-induct Ψ P α == iM(ν*xvec)⟨N⟩ P' avoiding: x arbitrary: M xvec
  N rule: semanticsInduct)
    case (cAlpha Ψ P α P' p x M xvec N)
    have S: set p ⊆ set(bn α) × set(bn(p · α)) by fact
    from ⟨p · α = iM(ν*xvec)⟨N⟩⟩ have (p · p · α) = p · (iM(ν*xvec)⟨N⟩)
  by (simp add: fresh-star-bij)
    with ⟨distinctPerm p⟩ have α = i(p · M)(ν*(p · xvec))⟨(p · N)⟩ by simp
    moreover from ⟨p · α = iM(ν*xvec)⟨N⟩⟩ ⟨x # xvec⟩ have x # (bn(p · α))
  by simp
    with ⟨(bn α) #* x⟩ ⟨x # xvec⟩ S have x # (p · xvec)
    by (fastforce dest: pt-fresh-bij1[OF pt-name-inst, OF at-name-inst, where
  pi=p and x=xvec])
    ultimately have x # (p · N) using ⟨x # P⟩ by (metis cAlpha)
    then have (p · x) # (p · p · N) by (simp add: pt-fresh-bij1[OF pt-name-inst,
  OF at-name-inst])
    with ⟨distinctPerm p⟩ ⟨bn(α) #* x⟩ ⟨x # (bn(p · α))⟩ S show ?case by simp
  next
    case cInput
    then show ?case by simp
  next
    case cBrInput
    then show ?case by simp
  next
    case cOutput
    then show ?case by (simp add: action.inject)
  next
    case cBrOutput
    then show ?case by (simp add: action.inject)
  next
    case cCase
    then show ?case by (auto simp add: action.inject dest: memFresh)
  next
    case cPar1
    then show ?case by simp
  next
    case cPar2
    then show ?case by simp
  next

```

```

    case cComm1
    then show ?case by simp
next
    case cComm2
    then show ?case by simp
next
    case cBrMerge
    then show ?case by simp
next
    case cBrComm1
    then show ?case by simp
next
    case cBrComm2
    then show ?case by simp
next
    case cBrClose
    then show ?case by simp
next
    case(cOpen  $\Psi$  P M xvec yvec N P' x y M' zvec N')
    then show ?case by simp
next
    case(cBrOpen  $\Psi$  P M xvec yvec N P' x y M' zvec N')
    from  $\langle \downarrow M(\nu*(xvec@x\#yvec)) \rangle \langle N \rangle = \downarrow M'(\nu*zvec) \langle N' \rangle$  have  $zvec = xvec@x\#yvec$ 
and  $N = N'$ 
    by(simp add: action.inject)+
    from  $\langle y \# (\nu x)P \rangle \langle x \# y \rangle$  have  $y \# P$  by(simp add: abs-fresh)
    moreover from  $\langle y \# zvec \rangle \langle zvec = xvec@x\#yvec \rangle$  have  $y \# (xvec@yvec)$ 
    by simp
    ultimately have  $y \# N$  by(fastforce intro!: cBrOpen)
    with  $\langle N = N' \rangle$  show ?case by simp
next
    case cScope
    then show ?case by(auto simp add: abs-fresh)
next
    case cBang
    then show ?case by simp
qed
qed

```

lemma *outputFreshDerivativeP*:

```

fixes  $\Psi$     :: 'b
and P      :: ('a, 'b, 'c) psi
and M      :: 'a
and xvec   :: name list
and N      :: 'a
and P'     :: ('a, 'b, 'c) psi
and x      :: name

```

assumes $\Psi \triangleright P \mapsto M(\nu*xvec) \langle N \rangle \prec P'$

```

and xvec #* M
and distinct xvec
and x # P
and x # xvec

shows x # P'
proof -
  note ⟨Ψ ▷ P ⟶ M(ν*xvec)⟨N⟩ < P'⟩
  moreover from ⟨xvec #* M⟩ have bn(M(ν*xvec)⟨N⟩) #* subject(M(ν*xvec)⟨N⟩)
  by simp
  moreover from ⟨distinct xvec⟩ have distinct(bn(M(ν*xvec)⟨N⟩)) by simp
  ultimately show x # P' using ⟨x # P⟩ ⟨x # xvec⟩
  proof (nominal-induct Ψ P α == M(ν*xvec)⟨N⟩ P' avoiding: x arbitrary: M xvec
    N rule: semanticsInduct)
    case (cAlpha Ψ P α P' p x M xvec N)
    have S: set p ⊆ set(bn α) × set(bn(p · α)) by fact
    from ⟨(p · α) = M(ν*xvec)⟨N⟩⟩ have (p · p · α) = p · (M(ν*xvec)⟨N⟩)
  by (simp add: fresh-star-bij)
    with ⟨distinctPerm p⟩ have α = (p · M)(ν*(p · xvec))⟨(p · N)⟩ by simp
    moreover from ⟨(p · α) = M(ν*xvec)⟨N⟩⟩ ⟨x # xvec⟩ have x # (bn(p · α)) by
  simp
    with ⟨(bn α) #* x⟩ ⟨x # xvec⟩ S have x # (p · xvec)
    by (fastforce dest: pt-fresh-bij1[OF pt-name-inst, OF at-name-inst, where
  pi=p and x=xvec])
    ultimately have x # P' using ⟨x # P⟩ by (metis cAlpha)
    then have (p · x) # (p · P') by (simp add: pt-fresh-bij1[OF pt-name-inst, OF
  at-name-inst])
    with ⟨distinctPerm p⟩ ⟨bn(α) #* x⟩ ⟨x # (bn(p · α))⟩ S show ?case by simp
  next
    case cInput
    then show ?case by simp
  next
    case cBrInput
    then show ?case by simp
  next
    case cOutput
    then show ?case by (simp add: action.inject)
  next
    case cBrOutput
    then show ?case by (simp add: action.inject)
  next
    case cCase
    then show ?case by (auto simp add: action.inject dest: memFresh)
  next
    case cPar1
    then show ?case by simp
  next
    case cPar2
    then show ?case by simp

```

```

next
  case cComm1
  then show ?case by simp
next
  case cComm2
  then show ?case by simp
next
  case cBrMerge
  then show ?case by simp
next
  case cBrComm1
  then show ?case by simp
next
  case cBrComm2
  then show ?case by simp
next
  case cBrClose
  then show ?case by simp
next
  case(cOpen  $\Psi$  P M xvec yvec N P' x y M' zvec N')
  from  $\langle M(\nu^*(xvec@x\#yvec))\rangle\langle N \rangle = M'(\nu^*zvec)\langle N' \rangle$  have zvec = xvec@x#yvec
  by(simp add: action.inject)
  from  $\langle y \# (\nu x)P \rangle \langle x \# y \rangle$  have y # P by(simp add: abs-fresh)
  moreover from  $\langle y \# zvec \rangle \langle zvec = xvec@x\#yvec \rangle$  have y # (xvec@yvec)
  by simp
  ultimately show y # P'
  by(fastforce intro!: cOpen)
next
  case cBrOpen
  then show ?case by simp
next
  case cScope
  then show ?case by(auto simp add: abs-fresh)
next
  case cBang
  then show ?case by simp
qed
qed

```

lemma *broutputFreshDerivativeP*:

```

fixes  $\Psi$     :: 'b
and P      :: ('a, 'b, 'c) psi
and M     :: 'a
and xvec  :: name list
and N     :: 'a
and P'    :: ('a, 'b, 'c) psi
and x     :: name

```

assumes $\Psi \triangleright P \mapsto_i M(\nu^*xvec)\langle N \rangle \prec P'$

```

and xvec #* M
and distinct xvec
and x # P
and x # xvec

shows x # P'
proof -
  note ⟨Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P'⟩
  moreover from ⟨xvec #* M⟩ have bn(iM(ν*xvec)⟨N⟩) #* subject(iM(ν*xvec)⟨N⟩)
  by simp
  moreover from ⟨distinct xvec⟩ have distinct(bn(iM(ν*xvec)⟨N⟩)) by simp
  ultimately show x # P' using ⟨x # P⟩ ⟨x # xvec⟩
  proof (nominal-induct Ψ P α==iM(ν*xvec)⟨N⟩ P' avoiding: x arbitrary: M xvec
    N rule: semanticsInduct)
    case (cAlpha Ψ P α P' p x M xvec N)
    have S: set p ⊆ set(bn α) × set(bn(p · α)) by fact
    from ⟨(p · α) = iM(ν*xvec)⟨N⟩⟩ have (p · p · α) = p · (iM(ν*xvec)⟨N⟩)
  by (simp add: fresh-star-bij)
    with ⟨distinctPerm p⟩ have α = i(p · M)(ν*(p · xvec))⟨(p · N)⟩ by simp
    moreover from ⟨(p · α) = iM(ν*xvec)⟨N⟩⟩ ⟨x # xvec⟩ have x # (bn(p · α))
  by simp
    with ⟨(bn α) #* x⟩ ⟨x # xvec⟩ S have x # (p · xvec)
    by (fastforce dest: pt-fresh-bij1[OF pt-name-inst, OF at-name-inst, where
      pi=p and x=xvec])
    ultimately have x # P' using ⟨x # P⟩ by (metis cAlpha)
    then have (p · x) # (p · P') by (simp add: pt-fresh-bij1[OF pt-name-inst, OF
      at-name-inst])
    with ⟨distinctPerm p⟩ ⟨bn(α) #* x⟩ ⟨x # (bn(p · α))⟩ S show ?case by simp
  next
  case cInput
  then show ?case by simp
  next
  case cBrInput
  then show ?case by simp
  next
  case cOutput
  then show ?case by (simp add: action.inject)
  next
  case cBrOutput
  then show ?case by (simp add: action.inject)
  next
  case cCase
  then show ?case by (auto simp add: action.inject dest: memFresh)
  next
  case cPar1
  then show ?case by simp
  next
  case cPar2
  then show ?case by simp

```

```

next
  case cComm1
  then show ?case by simp
next
  case cComm2
  then show ?case by simp
next
  case cBrMerge
  then show ?case by simp
next
  case (cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q \text{xvec} Q' A_Q x M' \text{zvec} N'$ )
  from  $\langle x \# (P \parallel Q) \rangle$  have  $x \# P$  and  $x \# Q$  by simp+

  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec Q' \rangle \langle \text{xvec} \#* M \rangle \langle \text{distinct xvec} \rangle \langle x \# Q \rangle \langle \text{xvec} \#* x \rangle$ 
  have  $x \# N$  by (simp add: broutputFreshDerivativeN)

  with  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \text{jM}(\nu) \langle N \rangle \prec P' \rangle \langle x \# P \rangle$  have  $x \# P'$  by (simp add: brinputFreshDerivative)

  then show ?case using cBrComm1 by simp
next
  case (cBrComm2  $\Psi \Psi_Q P M \text{xvec} N P' A_P \Psi_P Q Q' A_Q x M' \text{zvec} N'$ )
  from  $\langle x \# (P \parallel Q) \rangle$  have  $x \# P$  and  $x \# Q$  by simp+

  from  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec P' \rangle \langle \text{xvec} \#* M \rangle \langle \text{distinct xvec} \rangle \langle x \# P \rangle \langle \text{xvec} \#* x \rangle$ 
  have  $x \# N$  by (simp add: broutputFreshDerivativeN)

  with  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{jM}(\nu) \langle N \rangle \prec Q' \rangle \langle x \# Q \rangle$  have  $x \# Q'$  by (simp add: brinputFreshDerivative)

  then show ?case using cBrComm2 by simp
next
  case cBrClose
  then show ?case by simp
next
  case cOpen
  then show ?case by simp
next
  case (cBrOpen  $\Psi P M \text{xvec} \text{yvec} N P' x y M' \text{zvec} N'$ )
  from  $\langle \text{jM}(\nu * (\text{xvec} @ x \# \text{yvec})) \langle N \rangle = \text{jM}'(\nu * \text{zvec}) \langle N' \rangle \rangle$  have  $\text{zvec} = \text{xvec} @ x \# \text{yvec}$ 
  by (simp add: action.inject)
  from  $\langle y \# (\nu x) P \rangle \langle x \# y \rangle$  have  $y \# P$  by (simp add: abs-fresh)
  moreover from  $\langle y \# \text{zvec} \rangle \langle \text{zvec} = \text{xvec} @ x \# \text{yvec} \rangle$  have  $y \# (\text{xvec} @ \text{yvec})$ 
  by simp
  ultimately show  $y \# P'$ 
  by (fastforce intro: cBrOpen)
next

```

```

    case cScope
    then show ?case by(auto simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
  qed
qed

```

lemma *outputFreshDerivative:*

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $xvec$   :: name list
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $x$      :: name

```

```

assumes  $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $xvec \#* M$ 
and distinct  $xvec$ 
and  $x \# P$ 
and  $x \# xvec$ 

```

```

shows  $x \# N$ 
and  $x \# P'$ 
using assms
by(auto simp add: outputFreshDerivativeN outputFreshDerivativeP)

```

lemma *brouputFreshDerivative:*

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $xvec$   :: name list
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $x$      :: name

```

```

assumes  $\Psi \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $xvec \#* M$ 
and distinct  $xvec$ 
and  $x \# P$ 
and  $x \# xvec$ 

```

```

shows  $x \# N$ 
and  $x \# P'$ 
using assms
by(auto simp add: brouputFreshDerivativeN brouputFreshDerivativeP)

```

lemma *outputFreshChainDerivative:*

```

fixes  $\Psi$   :: 'b
and  $P$     :: ('a, 'b, 'c) psi
and  $M$     :: 'a
and  $xvec$  :: name list
and  $N$     :: 'a
and  $P'$    :: ('a, 'b, 'c) psi
and  $yvec$  :: name list

assumes  $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $xvec \#* M$ 
and distinct  $xvec$ 
and  $yvec \#* P$ 
and  $yvec \#* xvec$ 

shows  $yvec \#* N$ 
and  $yvec \#* P'$ 
using assms
by(induct yvec) (auto intro: outputFreshDerivative)

lemma broutputFreshChainDerivative:
fixes  $\Psi$   :: 'b
and  $P$     :: ('a, 'b, 'c) psi
and  $M$     :: 'a
and  $xvec$  :: name list
and  $N$     :: 'a
and  $P'$    :: ('a, 'b, 'c) psi
and  $yvec$  :: name list

assumes  $\Psi \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $xvec \#* M$ 
and distinct  $xvec$ 
and  $yvec \#* P$ 
and  $yvec \#* xvec$ 

shows  $yvec \#* N$ 
and  $yvec \#* P'$ 
using assms
by(induct yvec) (auto intro: broutputFreshDerivative)

lemma tauFreshDerivative:
fixes  $\Psi$   :: 'b
and  $P$     :: ('a, 'b, 'c) psi
and  $P'$    :: ('a, 'b, 'c) psi
and  $x$     :: name

assumes  $\Psi \triangleright P \mapsto \tau \prec P'$ 
and  $x \# P$ 

shows  $x \# P'$ 

```



```

proof –
  have  $bn(\tau) \#* \text{subject}(\tau)$  and  $\text{distinct}(bn(\tau))$  by simp+
  with  $\langle \Psi \triangleright P \mapsto \tau \prec P' \rangle$  show ?thesis using  $\langle x \# P \rangle$ 
  proof(nominal-induct  $\Psi P \alpha == (\tau :: ('a \text{ action})) P'$  avoiding: x rule: semantic-
sInduct)
    case cAlpha
    then show ?case by simp
  next
    case cInput
    then show ?case by simp
  next
    case cBrInput
    then show ?case by simp
  next
    case cOutput
    then show ?case by simp
  next
    case cBrOutput
    then show ?case by simp
  next
    case cCase
    then show ?case by (auto dest: memFresh)
  next
    case cPar1
    then show ?case by simp
  next
    case cPar2
    then show ?case by simp
  next
    case cComm1
    then show ?case
    by(auto dest: inputFreshDerivative outputFreshDerivative simp add: resChain-
Fresh)
  next
    case cComm2
    then show ?case
    by(auto dest: inputFreshDerivative outputFreshDerivative simp add: resChain-
Fresh)
  next
    case cBrMerge
    then show ?case by simp
  next
    case cBrComm1
    then show ?case by simp
  next
    case cBrComm2
    then show ?case by simp
  next
    case cBrClose

```

```

    then show ?case
    by(auto dest: brinputFreshDerivative broutputFreshDerivative simp add: resChain-
Fresh abs-fresh)
  next
    case cOpen
    then show ?case by simp
  next
    case cBrOpen
    then show ?case by simp
  next
    case cScope
    then show ?case by(simp add: abs-fresh)
  next
    case cBang
    then show ?case by simp
qed

```

lemma *tauFreshChainDerivative*:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $M$  :: 'a
and  $N$  :: 'a
and  $P'$  :: ('a, 'b, 'c) psi
and  $xvec$  :: name list

```

```

assumes  $\Psi \triangleright P \mapsto \tau \prec P'$ 
and  $xvec \#* P$ 

```

```

shows  $xvec \#* P'$ 
using assms
by(induct  $xvec$ ) (auto intro: tauFreshDerivative)

```

lemma *freeFreshDerivative*:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\alpha$  :: 'a action
and  $P'$  :: ('a, 'b, 'c) psi
and  $x$  :: name

```

```

assumes  $\Psi \triangleright P \mapsto \alpha \prec P'$ 
and  $bn \alpha \#* \text{subject } \alpha$ 
and  $\text{distinct}(bn \alpha)$ 
and  $x \# \alpha$ 
and  $x \# P$ 

```

```

shows  $x \# P'$ 
using assms
apply -

```

```

by(rule actionCases[where  $\alpha=\alpha$ ])
  (auto intro: inputFreshDerivative brinputFreshDerivative
    tauFreshDerivative
    outputFreshDerivative broutputFreshDerivative)

```

lemma *freeFreshChainDerivative*:

```

fixes  $\Psi$     :: 'b
and  $P$       :: ('a, 'b, 'c) psi
and  $\alpha$     :: 'a action
and  $P'$      :: ('a, 'b, 'c) psi
and  $xvec$    :: name list

```

assumes $\Psi \triangleright P \mapsto \alpha \prec P'$

```

and  $bn \ \alpha \ \#\# \ subject \ \alpha$ 
and  $distinct(bn \ \alpha)$ 
and  $xvec \ \#\# \ P$ 
and  $xvec \ \#\# \ \alpha$ 

```

shows $xvec \ \#\# \ P'$

```

using assms
by(auto intro: freeFreshDerivative simp add: fresh-star-def)

```

lemma *Input*:

```

fixes  $\Psi$     :: 'b
and  $M$       :: 'a
and  $K$       :: 'a
and  $xvec$    :: name list
and  $N$       :: 'a
and  $Tvec$    :: 'a list

```

assumes $\Psi \vdash M \leftrightarrow K$

```

and  $distinct \ xvec$ 
and  $set \ xvec \subseteq \supp \ N$ 
and  $length \ xvec = length \ Tvec$ 

```

shows $\Psi \triangleright M(\lambda * xvec \ N).P \mapsto K(\lambda [N[xvec::=Tvec]]) \prec P[xvec::=Tvec]$

proof –

```

obtain  $p$  where  $xvecFreshPsi$ :  $((p::name \ prm) \cdot (xvec::name \ list)) \ \#\# \ \Psi$ 
and  $xvecFreshM$ :  $(p \cdot xvec) \ \#\# \ M$ 
and  $xvecFreshN$ :  $(p \cdot xvec) \ \#\# \ N$ 
and  $xvecFreshK$ :  $(p \cdot xvec) \ \#\# \ K$ 
and  $xvecFreshTvec$ :  $(p \cdot xvec) \ \#\# \ Tvec$ 
and  $xvecFreshP$ :  $(p \cdot xvec) \ \#\# \ P$ 
and  $S$ :  $(set \ p) \subseteq (set \ xvec) \times (set \ (p \cdot xvec))$ 
and  $dp$ :  $distinctPerm \ p$ 

```

apply –

```

by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(\Psi, M, K, N, P, Tvec)$ ])
  (auto simp add: eqts fresh-star-prod)

```

note $\langle \Psi \vdash M \leftrightarrow K \rangle$

```

moreover from ⟨distinct xvec⟩ have distinct( $p \cdot xvec$ )
  by simp
moreover from ⟨ $(set\ xvec) \subseteq (supp\ N)$ ⟩ have  $(p \cdot (set\ xvec)) \subseteq (p \cdot (supp\ N))$ 
  by simp
then have  $set(p \cdot xvec) \subseteq supp(p \cdot N)$ 
  by(simp add: eqvts)
moreover from ⟨length xvec = length Tvec⟩ have length( $p \cdot xvec$ ) = length Tvec
  by simp
ultimately have  $\Psi \triangleright M(\lambda*(p \cdot xvec)\ (p \cdot N)).(p \cdot P) \mapsto K((p \cdot N)[(p \cdot xvec)::=Tvec]) \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$ 
  using xvecFreshPsi xvecFreshM xvecFreshK xvecFreshTvec
  by(metis cInput)
then show ?thesis using xvecFreshN xvecFreshP S ⟨length xvec = length Tvec⟩
dp
  by(auto simp add: inputChainAlpha' substTerm.renaming renaming)
qed

```

lemma *BrInput*:

```

fixes  $\Psi$    :: 'b
  and  $M$      :: 'a
  and  $K$      :: 'a
  and  $xvec$   :: name list
  and  $N$      :: 'a
  and  $Tvec$   :: 'a list

```

```

assumes  $\Psi \vdash K \succeq M$ 
  and distinct xvec
  and  $set\ xvec \subseteq supp\ N$ 
  and length xvec = length Tvec

```

shows $\Psi \triangleright M(\lambda*xvec\ N).P \mapsto ;K(N[xvec::=Tvec]) \prec P[xvec::=Tvec]$

proof –

```

obtain  $p$  where xvecFreshPsi:  $((p::name\ prm) \cdot (xvec::name\ list)) \#* \Psi$ 
  and xvecFreshM:  $(p \cdot xvec) \#* M$ 
  and xvecFreshN:  $(p \cdot xvec) \#* N$ 
  and xvecFreshK:  $(p \cdot xvec) \#* K$ 
  and xvecFreshTvec:  $(p \cdot xvec) \#* Tvec$ 
  and xvecFreshP:  $(p \cdot xvec) \#* P$ 
  and  $S: (set\ p) \subseteq (set\ xvec) \times (set(p \cdot xvec))$ 
  and dp: distinctPerm p

```

apply –

```

by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, M, K, N, P, Tvec)])
  (auto simp add: eqvts fresh-star-prod)

```

note ⟨ $\Psi \vdash K \succeq M$ ⟩

moreover from ⟨*distinct xvec*⟩ **have** *distinct*($p \cdot xvec$)

by *simp*

moreover from ⟨ $(set\ xvec) \subseteq (supp\ N)$ ⟩ **have** $(p \cdot (set\ xvec)) \subseteq (p \cdot (supp\ N))$

by *simp*

then have $set(p \cdot xvec) \subseteq supp(p \cdot N)$

```

  by(simp add: eqvts)
  moreover from ⟨length xvec = length Tvec⟩ have length(p · xvec) = length Tvec
  by simp
  ultimately have  $\Psi \triangleright M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot P) \mapsto_i K((p \cdot N)[(p \cdot xvec)::=Tvec]) \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$ 
  using xvecFreshPsi xvecFreshM xvecFreshK xvecFreshTvec
  by(metis cBrInput)
  then show ?thesis using xvecFreshN xvecFreshP S ⟨length xvec = length Tvec⟩
  dp
  by(auto simp add: inputChainAlpha' substTerm.renaming renaming)
qed

```

lemma residualAlpha:

```

  fixes p :: name prm
  and  $\alpha$  :: 'a action
  and P :: ('a, 'b, 'c) psi

```

```

assumes bn(p ·  $\alpha$ )  $\#$ * object  $\alpha$ 
and bn(p ·  $\alpha$ )  $\#$ * P
and bn  $\alpha$   $\#$ * subject  $\alpha$ 
and bn(p ·  $\alpha$ )  $\#$ * subject  $\alpha$ 
and set p  $\subseteq$  set(bn  $\alpha$ )  $\times$  set(bn(p ·  $\alpha$ ))

```

```

shows  $\alpha \prec P = (p \cdot \alpha) \prec (p \cdot P)$ 
using assms
apply -
apply(rule actionCases[where  $\alpha=\alpha$ ])
  apply(simp only: eqvts bn.simps)
  apply simp
  apply(simp only: eqvts bn.simps)
  apply simp
  apply simp
  apply(simp add: boundOutputChainAlpha'' residualInject)
  apply simp
  apply(simp add: boundOutputChainAlpha'' residualInject)
by simp

```

lemma Par1:

```

  fixes  $\Psi$  :: 'b
  and  $\Psi_Q$  :: 'b
  and P :: ('a, 'b, 'c) psi
  and  $\alpha$  :: 'a action
  and P' :: ('a, 'b, 'c) psi
  and  $A_Q$  :: name list
  and Q :: ('a, 'b, 'c) psi

```

```

assumes Trans:  $\Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'$ 
and extractFrame Q = ⟨ $A_Q$ ,  $\Psi_Q$ ⟩
and bn  $\alpha$   $\#$ * Q

```

```

and  $A_Q \#* \Psi$ 
and  $A_Q \#* P$ 
and  $A_Q \#* \alpha$ 

shows  $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P' \parallel Q)$ 
proof –
{
  fix  $\Psi \quad :: 'b$ 
  and  $\Psi_Q \quad :: 'b$ 
  and  $P \quad \quad :: ('a, 'b, 'c) \text{ psi}$ 
  and  $\alpha \quad \quad :: 'a \text{ action}$ 
  and  $P' \quad \quad :: ('a, 'b, 'c) \text{ psi}$ 
  and  $A_Q \quad \quad :: \text{ name list}$ 
  and  $Q \quad \quad :: ('a, 'b, 'c) \text{ psi}$ 

  assume  $\Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'$ 
  and  $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ 
  and  $\text{bn } \alpha \#* Q$ 
  and  $\text{bn } \alpha \#* \text{subject } \alpha$ 
  and  $A_Q \#* \Psi$ 
  and  $A_Q \#* P$ 
  and  $A_Q \#* \alpha$ 
  and  $\text{distinct } A_Q$ 

  have  $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P' \parallel Q)$ 
  proof –
  from  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle$  have  $\text{distinct}(\text{bn } \alpha)$  by(rule boundOutput-Distinct)
  obtain  $q::\text{name prm}$  where  $\text{bn}(q \cdot \alpha) \#* \Psi$  and  $\text{bn}(q \cdot \alpha) \#* P$  and  $\text{bn}(q \cdot \alpha) \#* Q$  and  $\text{bn}(q \cdot \alpha) \#* \alpha$ 
  and  $\text{bn}(q \cdot \alpha) \#* A_Q$  and  $\text{bn}(q \cdot \alpha) \#* P'$  and  $\text{bn}(q \cdot \alpha) \#* \Psi_Q$ 
  and  $Sq: (\text{set } q) \subseteq (\text{set } (\text{bn } \alpha)) \times (\text{set } (\text{bn}(q \cdot \alpha)))$ 
  apply –
  by(rule name-list-avoiding[where  $\text{vvec}=\text{bn } \alpha$  and  $c=(\Psi, P, Q, \alpha, A_Q, \Psi_Q, P')$ ]) (auto simp add: eqvts)
  obtain  $p::\text{name prm}$  where  $(p \cdot A_Q) \#* \Psi$  and  $(p \cdot A_Q) \#* P$  and  $(p \cdot A_Q) \#* Q$  and  $(p \cdot A_Q) \#* \alpha$ 
  and  $(p \cdot A_Q) \#* \alpha$  and  $(p \cdot A_Q) \#* (q \cdot \alpha)$  and  $(p \cdot A_Q) \#* P'$ 
  and  $(p \cdot A_Q) \#* (q \cdot P')$  and  $(p \cdot A_Q) \#* \Psi_Q$  and  $S_p: (\text{set } p) \subseteq (\text{set } A_Q) \times (\text{set } (p \cdot A_Q))$ 
  apply –
  by(rule name-list-avoiding[where  $\text{vvec}=A_Q$  and  $c=(\Psi, P, Q, \alpha, \text{bn } \alpha, q \cdot \alpha, P', (q \cdot P'), \Psi_Q)$ ]) auto
  from  $\langle \text{distinct}(\text{bn } \alpha) \rangle$  have  $\text{distinct}(\text{bn}(q \cdot \alpha))$ 
  by – (rule actionCases[where  $\alpha=\alpha$ ], auto simp add: eqvts)
  from  $\langle A_Q \#* \alpha \rangle \langle \text{bn}(q \cdot \alpha) \#* A_Q \rangle Sq$  have  $A_Q \#* (q \cdot \alpha)$ 
  apply –
  apply(rule actionCases[where  $\alpha=\alpha$ ])
  apply(simp only: bn.simps eqvts, simp)

```

```

    apply(simp only: bn.simps eqvts, simp)
    apply(simp add: freshChainSimps)
    apply(simp add: freshChainSimps)
  by simp
  from ⟨bn α #* subject α⟩ have (q · (bn α)) #* (q · (subject α))
    by(simp add: fresh-star-bij)
  then have bn(q · α) #* subject(q · α) by(simp add: eqvts)
  from ⟨Ψ ⊗ Ψ_Q ▷ P ⟶ α < P'⟩ ⟨bn(q · α) #* α⟩ ⟨bn(q · α) #* P'⟩ ⟨bn α #*
(subject α)⟩ Sq
  have Trans: Ψ ⊗ Ψ_Q ▷ P ⟶ (q · α) < (q · P')
    by(force simp add: residualAlpha)
  then have A_Q #* (q · P') using ⟨bn(q · α) #* subject(q · α)⟩ ⟨distinct(bn(q
· α))⟩ ⟨A_Q #* P⟩ ⟨A_Q #* (q · α)⟩
    by(auto intro: freeFreshChainDerivative)
  from Trans have (p · (Ψ ⊗ Ψ_Q)) ▷ (p · P) ⟶ p · ((q · α) < (q · P'))
    by(rule semantics.eqvt)
  with ⟨A_Q #* Ψ⟩ ⟨A_Q #* P⟩ ⟨A_Q #* (q · α)⟩ ⟨(p · A_Q) #* (q · α)⟩ ⟨A_Q #* (q ·
P')⟩
    ⟨(p · A_Q) #* Ψ⟩ ⟨(p · A_Q) #* P⟩ ⟨(p · A_Q) #* (q · P')⟩ Sp
  have Ψ ⊗ (p · Ψ_Q) ▷ P ⟶ (q · α) < (q · P') by(simp add: eqvts)
  moreover from ⟨extractFrame Q = ⟨A_Q, Ψ_Q⟩⟩ ⟨(p · A_Q) #* Ψ_Q⟩ Sp have
extractFrame Q = ⟨(p · A_Q), (p · Ψ_Q)⟩
    by(simp add: frameChainAlpha' eqvts)
  moreover from ⟨(bn(q · α)) #* Ψ_Q⟩ ⟨(bn(q · α)) #* A_Q⟩ ⟨(p · A_Q) #* (q ·
α)⟩ Sp
    have (bn(q · α)) #* (p · Ψ_Q)
      by(simp add: freshAlphaPerm)
  moreover from ⟨distinct A_Q⟩ have distinct(p · A_Q) by simp
  ultimately have Ψ ▷ P || Q ⟶ (q · α) < ((q · P') || Q)
    using ⟨(p · A_Q) #* P⟩ ⟨(p · A_Q) #* Q⟩ ⟨(p · A_Q) #* Ψ⟩ ⟨(p · A_Q) #* (q · α)⟩
      ⟨(p · A_Q) #* (q · P')⟩ ⟨(bn(q · α)) #* Ψ⟩ ⟨(bn(q · α)) #* Q⟩ ⟨(bn(q · α))
#* P⟩
      ⟨(bn(q · α)) #* (subject (q · α))⟩ ⟨distinct(bn(q · α))⟩
    by(metis cPar1)

  then show ?thesis using ⟨bn(q · α) #* α⟩ ⟨bn(q · α) #* P'⟩ ⟨bn α #* subject
α⟩ ⟨bn(q · α) #* Q⟩ ⟨bn α #* Q⟩ Sq
    by(force simp add: residualAlpha)
  qed
}
note Goal = this
from ⟨extractFrame Q = ⟨A_Q, Ψ_Q⟩⟩ ⟨A_Q #* Ψ⟩ ⟨A_Q #* P⟩ ⟨A_Q #* α⟩
  obtain A_Q' where FrQ: extractFrame Q = ⟨A_Q', Ψ_Q⟩ and distinct A_Q' and
A_Q' #* Ψ and A_Q' #* P and A_Q' #* α
  apply -
  by(rule distinctFrame[where C=(Ψ, P, α)]) auto
show ?thesis
proof(induct rule: actionCases[where α=α])
  case(cInput M N)

```

```

from Trans FrQ  $\langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle A_Q' \#* \alpha \rangle \langle \text{distinct } A_Q' \rangle \langle \text{bn } \alpha \#* Q \rangle$ 
show ?case using  $\langle \alpha = M(\downarrow N) \rangle$  by(force intro: Goal)
next
  case(cBrInput M N)
  from Trans FrQ  $\langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle A_Q' \#* \alpha \rangle \langle \text{distinct } A_Q' \rangle \langle \text{bn } \alpha \#* Q \rangle$ 
  show ?case using  $\langle \alpha = \downarrow_i M(\downarrow N) \rangle$  by(force intro: Goal)
next
  case cTau
  from Trans FrQ  $\langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle A_Q' \#* \alpha \rangle \langle \text{distinct } A_Q' \rangle \langle \text{bn } \alpha \#* Q \rangle$ 
  show ?case using  $\langle \alpha = \tau \rangle$  by(force intro: Goal)
next
  case(cOutput M xvec N)
  from  $\langle \alpha = M(\downarrow \nu * xvec) \rangle \langle N \rangle \langle A_Q' \#* \alpha \rangle \langle \text{bn } \alpha \#* Q \rangle$  have  $xvec \#* A_Q'$  and  $xvec \#* Q$ 
  by simp+
  obtain  $p$  where  $(p \cdot xvec) \#* N$  and  $(p \cdot xvec) \#* P'$  and  $(p \cdot xvec) \#* Q$ 
  and  $(p \cdot xvec) \#* M$  and  $(p \cdot xvec) \#* A_Q'$ 
  and  $S: \text{set } p \subseteq \text{set } xvec \times \text{set}(p \cdot xvec)$ 
  apply –
  by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(N, P', Q, M, A_Q')$ ])
auto
  from Trans  $\langle \alpha = M(\downarrow \nu * xvec) \rangle \langle N \rangle$  have  $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\downarrow \nu * xvec) \langle N \rangle \prec P'$ 
by simp
  with  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle S$ 
  have  $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\downarrow \nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot P')$ 
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
  moreover from  $\langle xvec \#* A_Q' \rangle \langle (p \cdot xvec) \#* A_Q' \rangle \langle A_Q' \#* \alpha \rangle S$ 
  have  $A_Q' \#* (p \cdot \alpha)$  by(simp add: freshChainSimps del: actionFreshChain)
  ultimately have  $\Psi \triangleright P \parallel Q \mapsto M(\downarrow \nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot P') \parallel Q$ 
  using FrQ  $\langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle \text{distinct } A_Q' \rangle \langle (p \cdot xvec) \#* Q \rangle \langle A_Q' \#* \alpha \rangle$ 
   $\langle (p \cdot xvec) \#* M \rangle \langle \alpha = M(\downarrow \nu * xvec) \rangle \langle N \rangle$ 
  by(force intro: Goal)
  with  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle \langle (p \cdot xvec) \#* Q \rangle \langle xvec \#* Q \rangle S \langle \alpha = M(\downarrow \nu * xvec) \rangle \langle N \rangle$ 
  show ?case
  by(simp add: boundOutputChainAlpha'' eqvts create-residual.simps)
next
  case(cBrOutput M xvec N)
  from  $\langle \alpha = \downarrow_i M(\downarrow \nu * xvec) \rangle \langle N \rangle \langle A_Q' \#* \alpha \rangle \langle \text{bn } \alpha \#* Q \rangle$  have  $xvec \#* A_Q'$  and  $xvec \#* Q$ 
  by simp+
  obtain  $p$  where  $(p \cdot xvec) \#* N$  and  $(p \cdot xvec) \#* P'$  and  $(p \cdot xvec) \#* Q$ 
  and  $(p \cdot xvec) \#* M$  and  $(p \cdot xvec) \#* A_Q'$ 
  and  $S: \text{set } p \subseteq \text{set } xvec \times \text{set}(p \cdot xvec)$ 
  by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(N, P', Q, M, A_Q')$ ])
auto
  from Trans  $\langle \alpha = \downarrow_i M(\downarrow \nu * xvec) \rangle \langle N \rangle$  have  $\Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow_i M(\downarrow \nu * xvec) \langle N \rangle \prec P'$ 
by simp
  with  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle S$ 

```



```

have  $\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM} (\nu^*(p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot P')$ 
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
moreover from  $\langle xvec \#* A_{Q'} \rangle \langle (p \cdot xvec) \#* A_{Q'} \rangle \langle A_{Q'} \#* \alpha \rangle S$ 
have  $A_{Q'} \#* (p \cdot \alpha)$  by(simp add: freshChainSimps del: actionFreshChain)
ultimately have  $\Psi \triangleright P \parallel Q \mapsto_{iM} (\nu^*(p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot P') \parallel Q$ 
  using FrQ  $\langle A_{Q'} \#* \Psi \rangle \langle A_{Q'} \#* P \rangle \langle distinct A_{Q'} \rangle \langle (p \cdot xvec) \#* Q \rangle \langle A_{Q'} \#* \alpha \rangle$ 
   $\langle (p \cdot xvec) \#* M \rangle \langle \alpha =_{iM} (\nu^* xvec) \langle N \rangle \rangle$ 
  by(force intro: Goal)
with  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle \langle (p \cdot xvec) \#* Q \rangle \langle xvec \#* Q \rangle S \langle \alpha =_{iM} (\nu^* xvec) \langle N \rangle \rangle$ 
show ?case
  by(simp add: boundOutputChainAlpha'' eqvts create-residual.simps)
qed
qed

```

lemma *Par2*:

```

fixes  $\Psi$    :: 'b
and  $\Psi_P$   :: 'b
and  $Q$      :: ('a, 'b, 'c) psi
and  $\alpha$    :: 'a action
and  $Q'$    :: ('a, 'b, 'c) psi
and  $A_P$    :: name list
and  $P$      :: ('a, 'b, 'c) psi

```

```

assumes Trans:  $\Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'$ 
and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
and  $bn \alpha \#* P$ 
and  $A_P \#* \Psi$ 
and  $A_P \#* Q$ 
and  $A_P \#* \alpha$ 

```

```

shows  $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$ 
proof -

```

```

{
  fix  $\Psi$    :: 'b
  and  $\Psi_P$  :: 'b
  and  $Q$     :: ('a, 'b, 'c) psi
  and  $\alpha$   :: 'a action
  and  $Q'$   :: ('a, 'b, 'c) psi
  and  $A_P$   :: name list
  and  $P$     :: ('a, 'b, 'c) psi

```

```

assume  $\Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'$ 
and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
and  $bn \alpha \#* P$ 
and  $bn \alpha \#* subject \alpha$ 
and  $A_P \#* \Psi$ 
and  $A_P \#* Q$ 
and  $A_P \#* \alpha$ 

```

```

and distinct  $A_P$ 

have  $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$ 
proof –
  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle$  have distinct( $bn \ \alpha$ ) by(rule boundOutput-Distinct)
  obtain  $q::name \ prn$  where  $bn(q \cdot \alpha) \#* \Psi$  and  $bn(q \cdot \alpha) \#* P$  and  $bn(q \cdot \alpha) \#* Q$  and  $bn(q \cdot \alpha) \#* \alpha$ 
    and  $bn(q \cdot \alpha) \#* A_P$  and  $bn(q \cdot \alpha) \#* Q'$  and  $bn(q \cdot \alpha) \#* \Psi_P$ 
    and  $Sq: (set \ q) \subseteq (set \ (bn \ \alpha)) \times (set \ (bn(q \cdot \alpha)))$ 
    by(rule name-list-avoiding[where  $xvec=bn \ \alpha$  and  $c=(\Psi, P, Q, \alpha, A_P, \Psi_P, Q')$ ]) (auto simp add: eqvts)
  obtain  $p::name \ prm$  where  $(p \cdot A_P) \#* \Psi$  and  $(p \cdot A_P) \#* P$  and  $(p \cdot A_P) \#* Q$  and  $(p \cdot A_P) \#* \alpha$ 
    and  $(p \cdot A_P) \#* \alpha$  and  $(p \cdot A_P) \#* (q \cdot \alpha)$  and  $(p \cdot A_P) \#* Q'$ 
    and  $(p \cdot A_P) \#* (q \cdot Q')$  and  $(p \cdot A_P) \#* \Psi_P$ 
    and  $Sp: (set \ p) \subseteq (set \ A_P) \times (set \ (p \cdot A_P))$ 
    by(rule name-list-avoiding[where  $xvec=A_P$  and  $c=(\Psi, P, Q, \alpha, q \cdot \alpha, Q', (q \cdot Q'), \Psi_P)$ ]) auto
  from  $\langle distinct(bn \ \alpha) \rangle$  have distinct( $bn(q \cdot \alpha)$ )
    apply –
    by(rule actionCases[where  $\alpha=\alpha$ ]) (auto simp add: eqvts)
  from  $\langle A_P \#* \alpha \rangle \langle bn(q \cdot \alpha) \#* A_P \rangle Sq$  have  $A_P \#* (q \cdot \alpha)$ 
    apply –
    apply(rule actionCases[where  $\alpha=\alpha$ ])
      apply(simp only: bn.simps eqvts, simp)
      apply(simp only: bn.simps eqvts, simp)
      apply(simp add: freshChainSimps)
      apply(simp add: freshChainSimps)
    by simp
  from  $\langle bn \ \alpha \#* subject \ \alpha \rangle$  have  $(q \cdot (bn \ \alpha)) \#* (q \cdot (subject \ \alpha))$ 
    by(simp add: fresh-star-bij)
  then have  $bn(q \cdot \alpha) \#* subject(q \cdot \alpha)$  by(simp add: eqvts)
  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle \langle bn(q \cdot \alpha) \#* \alpha \rangle \langle bn(q \cdot \alpha) \#* Q' \rangle \langle bn \ \alpha \#* subject \ \alpha \rangle Sq$ 
    have  $Trans: \Psi \otimes \Psi_P \triangleright Q \mapsto (q \cdot \alpha) \prec (q \cdot Q')$ 
      by(force simp add: residualAlpha)
    then have  $A_P \#* (q \cdot Q')$  using  $\langle bn(q \cdot \alpha) \#* subject(q \cdot \alpha) \rangle \langle distinct(bn(q \cdot \alpha)) \rangle \langle A_P \#* Q \rangle \langle A_P \#* (q \cdot \alpha) \rangle$ 
      by(auto intro: freeFreshChainDerivative)
    from  $Trans$  have  $(p \cdot (\Psi \otimes \Psi_P)) \triangleright (p \cdot Q) \mapsto p \cdot ((q \cdot \alpha) \prec (q \cdot Q'))$ 
      by(rule semantics.eqvt)
    with  $\langle A_P \#* \Psi \rangle \langle A_P \#* Q \rangle \langle A_P \#* (q \cdot \alpha) \rangle \langle (p \cdot A_P) \#* (q \cdot \alpha) \rangle \langle A_P \#* (q \cdot Q') \rangle$ 
       $\langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (p \cdot A_P) \#* (q \cdot Q') \rangle Sp$ 
    have  $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (q \cdot \alpha) \prec (q \cdot Q')$  by(simp add: eqvts)
    moreover from  $\langle extractFrame \ P = \langle A_P, \Psi_P \rangle \rangle \langle (p \cdot A_P) \#* \Psi_P \rangle Sp$  have
extractFrame  $P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$ 
      by(simp add: frameChainAlpha' eqvts)

```

```

    moreover from ⟨bn(q · α)⟩ #* Ψ_P ⟨bn(q · α)⟩ #* A_P ⟨p · A_P⟩ #* (q ·
α)⟩ Sp
    have (bn(q · α)⟩ #* (p · Ψ_P)
      by(simp add: freshAlphaPerm)
    moreover from ⟨distinct A_P⟩ have distinct(p · A_P) by simp
    ultimately have Ψ ▷ P ∥ Q ⟶ (q · α) ◁ (P ∥ (q · Q'))
      using ⟨p · A_P⟩ #* P ⟨p · A_P⟩ #* Q ⟨p · A_P⟩ #* Ψ ⟨p · A_P⟩ #* (q · α)
        ⟨p · A_P⟩ #* (q · Q') ⟨bn(q · α)⟩ #* Ψ ⟨bn(q · α)⟩ #* Q ⟨bn(q · α)⟩
#* P
        ⟨bn(q · α)⟩ #* (subject (q · α)) ⟨distinct(bn(q · α))⟩
      by(metis cPar2)

    then show ?thesis using ⟨bn(q · α)⟩ #* α ⟨bn(q · α)⟩ #* Q' ⟨bn α⟩ #* subject
α) ⟨bn(q · α)⟩ #* P ⟨bn α⟩ #* P Sq
      by(force simp add: residualAlpha)
    qed
  }
  note Goal = this
  from ⟨extractFrame P = ⟨A_P, Ψ_P⟩⟩ ⟨A_P⟩ #* Ψ ⟨A_P⟩ #* Q' ⟨A_P⟩ #* α
  obtain A_P' where FrP: extractFrame P = ⟨A_P', Ψ_P⟩ and distinct A_P' and
A_P' #* Ψ and A_P' #* Q and A_P' #* α
  apply -
  by(rule distinctFrame[where C=(Ψ, Q, α)]) auto
  show ?thesis
  proof(induct rule: actionCases[where α=α])
    case(cInput M N)
    from Trans FrP ⟨A_P'⟩ #* Ψ ⟨A_P'⟩ #* Q' ⟨A_P'⟩ #* α ⟨distinct A_P'⟩ ⟨bn α⟩ #* P
    show ?case using ⟨α = M(N)⟩ by(force intro: Goal)
  next
    case(cBrInput M N)
    from Trans FrP ⟨A_P'⟩ #* Ψ ⟨A_P'⟩ #* Q' ⟨A_P'⟩ #* α ⟨distinct A_P'⟩ ⟨bn α⟩ #* P
    show ?case using ⟨α = iM(N)⟩ by(force intro: Goal)
  next
    case cTau
    from Trans FrP ⟨A_P'⟩ #* Ψ ⟨A_P'⟩ #* Q' ⟨A_P'⟩ #* α ⟨distinct A_P'⟩ ⟨bn α⟩ #* P
    show ?case using ⟨α = τ⟩ by(force intro: Goal)
  next
    case(cOutput M xvec N)
    from ⟨α = M(ν*xvec)(N)⟩ ⟨A_P'⟩ #* α ⟨bn α⟩ #* P have xvec #* A_P' and xvec
#* P
      by simp+
    obtain p where (p · xvec) #* N and (p · xvec) #* Q' and (p · xvec) #* P
      and (p · xvec) #* M and (p · xvec) #* A_P'
      and S: set p ⊆ set xvec × set(p · xvec)
      by(rule name-list-avoiding[where xvec=xvec and c=(N, Q', P, M, A_P')])
    auto
    from Trans ⟨α=M(ν*xvec)(N)⟩ have Ψ ⊗ Ψ_P ▷ Q ⟶ M(ν*xvec)(N) ◁ Q'
  by simp
  with ⟨(p · xvec) #* N⟩ ⟨(p · xvec) #* Q'⟩ S

```

```

have  $\Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu^*(p \cdot \text{vec})) \langle (p \cdot N) \rangle \prec (p \cdot Q')$ 
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
moreover from  $\langle \text{vec} \#* A_{P'} \rangle \langle (p \cdot \text{vec}) \#* A_{P'} \rangle \langle A_{P'} \#* \alpha \rangle S$ 
have  $A_{P'} \#* (p \cdot \alpha)$  by(simp add: freshChainSimps del: actionFreshChain)
ultimately have  $\Psi \triangleright P \parallel Q \mapsto M(\nu^*(p \cdot \text{vec})) \langle (p \cdot N) \rangle \prec P \parallel (p \cdot Q')$ 
  using FrP  $\langle A_{P'} \#* \Psi \rangle \langle A_{P'} \#* Q \rangle \langle \text{distinct } A_{P'} \rangle \langle (p \cdot \text{vec}) \#* P \rangle \langle A_{P'} \#* \alpha \rangle$ 
   $\langle (p \cdot \text{vec}) \#* M \rangle \langle \alpha = M(\nu^* \text{vec}) \langle N \rangle \rangle$ 
  by(force intro: Goal)
with  $\langle (p \cdot \text{vec}) \#* N \rangle \langle (p \cdot \text{vec}) \#* Q' \rangle \langle (p \cdot \text{vec}) \#* P \rangle \langle \text{vec} \#* P \rangle S \langle \alpha =$ 
 $M(\nu^* \text{vec}) \langle N \rangle \rangle$ 
  show ?case
  by(simp add: boundOutputChainAlpha'' eqvts create-residual.simps)
next
case(cBrOutput M vec N)
  from  $\langle \alpha = \text{i}M(\nu^* \text{vec}) \langle N \rangle \rangle \langle A_{P'} \#* \alpha \rangle \langle \text{bn } \alpha \#* P \rangle$  have  $\text{vec} \#* A_{P'}$  and
 $\text{vec} \#* P$ 
  by simp+
obtain  $p$  where  $(p \cdot \text{vec}) \#* N$  and  $(p \cdot \text{vec}) \#* Q'$  and  $(p \cdot \text{vec}) \#* P$ 
and  $(p \cdot \text{vec}) \#* M$  and  $(p \cdot \text{vec}) \#* A_{P'}$ 
and  $S$ : set  $p \subseteq \text{set } \text{vec} \times \text{set}(p \cdot \text{vec})$ 
  by(rule name-list-avoiding[where vec=vec and c=(N, Q', P, M, A_{P'})])
auto
from Trans  $\langle \alpha = \text{i}M(\nu^* \text{vec}) \langle N \rangle \rangle$  have  $\Psi \otimes \Psi_P \triangleright Q \mapsto \text{i}M(\nu^* \text{vec}) \langle N \rangle \prec Q'$ 
by simp
with  $\langle (p \cdot \text{vec}) \#* N \rangle \langle (p \cdot \text{vec}) \#* Q' \rangle S$ 
have  $\Psi \otimes \Psi_P \triangleright Q \mapsto \text{i}M(\nu^*(p \cdot \text{vec})) \langle (p \cdot N) \rangle \prec (p \cdot Q')$ 
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
moreover from  $\langle \text{vec} \#* A_{P'} \rangle \langle (p \cdot \text{vec}) \#* A_{P'} \rangle \langle A_{P'} \#* \alpha \rangle S$ 
have  $A_{P'} \#* (p \cdot \alpha)$  by(simp add: freshChainSimps del: actionFreshChain)
ultimately have  $\Psi \triangleright P \parallel Q \mapsto \text{i}M(\nu^*(p \cdot \text{vec})) \langle (p \cdot N) \rangle \prec P \parallel (p \cdot Q')$ 
  using FrP  $\langle A_{P'} \#* \Psi \rangle \langle A_{P'} \#* Q \rangle \langle \text{distinct } A_{P'} \rangle \langle (p \cdot \text{vec}) \#* P \rangle \langle A_{P'} \#* \alpha \rangle$ 
   $\langle (p \cdot \text{vec}) \#* M \rangle \langle \alpha = \text{i}M(\nu^* \text{vec}) \langle N \rangle \rangle$ 
  by(force intro: Goal)
with  $\langle (p \cdot \text{vec}) \#* N \rangle \langle (p \cdot \text{vec}) \#* Q' \rangle \langle (p \cdot \text{vec}) \#* P \rangle \langle \text{vec} \#* P \rangle S \langle \alpha =$ 
 $\text{i}M(\nu^* \text{vec}) \langle N \rangle \rangle$ 
  show ?case
  by(simp add: boundOutputChainAlpha'' eqvts create-residual.simps)
qed
qed

```

lemma *Open*:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $M$  :: 'a
and  $\text{vec}$  :: name list
and  $\text{yvec}$  :: name list
and  $N$  :: 'a
and  $P'$  :: ('a, 'b, 'c) psi
and  $x$  :: name

```

assumes $\text{Trans}: \Psi \triangleright P \mapsto M(\nu^*(\text{vec}@yvec)) \langle N \rangle \prec P'$
and $x \in \text{supp } N$
and $x \# \Psi$
and $x \# M$
and $x \# \text{vec}$
and $x \# yvec$

shows $\Psi \triangleright (\nu x)P \mapsto M(\nu^*(\text{vec}@x\#yvec)) \langle N \rangle \prec P'$
proof –

from Trans **have** $\text{distinct}(\text{vec}@yvec)$ **by** $(\text{force dest: boundOutputDistinct})$
then have $\text{vec} \#* yvec$ **by** (induct vec) *auto*

obtain p **where** $(p \cdot yvec) \#* \Psi$ **and** $(p \cdot yvec) \#* P$ **and** $(p \cdot yvec) \#* M$
and $(p \cdot yvec) \#* yvec$ **and** $(p \cdot yvec) \#* N$ **and** $(p \cdot yvec) \#* P'$
and $x \# (p \cdot yvec)$ **and** $(p \cdot yvec) \#* \text{vec}$
and $S_p: (\text{set } p) \subseteq (\text{set } yvec) \times (\text{set}(p \cdot yvec))$
by $(\text{rule name-list-avoiding}[\text{where } \text{vec}=yvec \text{ and } c=(\Psi, P, M, \text{vec}, yvec, N, P', x)])$
(auto simp add: eqvts fresh-star-prod)

obtain q **where** $(q \cdot \text{vec}) \#* \Psi$ **and** $(q \cdot \text{vec}) \#* P$ **and** $(q \cdot \text{vec}) \#* M$
and $(q \cdot \text{vec}) \#* \text{vec}$ **and** $(q \cdot \text{vec}) \#* N$ **and** $(q \cdot \text{vec}) \#* P'$
and $x \# (q \cdot \text{vec})$ **and** $(q \cdot \text{vec}) \#* yvec$
and $(q \cdot \text{vec}) \#* p$ **and** $(q \cdot \text{vec}) \#* (p \cdot yvec)$
and $S_q: (\text{set } q) \subseteq (\text{set } \text{vec}) \times (\text{set}(q \cdot \text{vec}))$
by $(\text{rule name-list-avoiding}[\text{where } \text{vec}=\text{vec} \text{ and } c=(\Psi, P, M, \text{vec}, yvec, p \cdot yvec, N, P', x, p)])$
(auto simp add: eqvts fresh-star-prod)

note $\langle \Psi \triangleright P \mapsto M(\nu^*(\text{vec}@yvec)) \rangle \langle N \rangle \prec P'$
moreover from $\langle (p \cdot yvec) \#* N \rangle \langle (q \cdot \text{vec}) \#* N \rangle \langle \text{vec} \#* yvec \rangle \langle (q \cdot \text{vec}) \#* yvec \rangle \langle (q \cdot \text{vec}) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* \text{vec} \rangle S_p S_q$
have $((p@q) \cdot (\text{vec} @ yvec)) \#* N$
apply $(\text{simp only: eqvts})$
apply $(\text{simp only: pt2[OF pt-name-inst]})$
by *simp*

moreover from $\langle (p \cdot yvec) \#* P' \rangle \langle (q \cdot \text{vec}) \#* P' \rangle \langle \text{vec} \#* yvec \rangle \langle (q \cdot \text{vec}) \#* yvec \rangle \langle (q \cdot \text{vec}) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* \text{vec} \rangle S_p S_q$
have $((p@q) \cdot (\text{vec} @ yvec)) \#* P'$ **by** $(\text{simp del: freshAlphaPerm add: eqvts pt2[OF pt-name-inst]})$

moreover from $S_p S_q \langle \text{vec} \#* yvec \rangle \langle (q \cdot \text{vec}) \#* yvec \rangle \langle (q \cdot \text{vec}) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* \text{vec} \rangle$
have $S_{pq}: \text{set}(p@q) \subseteq \text{set}(\text{vec}@yvec) \times \text{set}((p@q) \cdot (\text{vec}@yvec))$
by $(\text{simp add: pt2[OF pt-name-inst] eqvts blast})$

ultimately have $\Psi \triangleright P \mapsto M(\nu^*((p@q) \cdot (\text{vec}@yvec))) \langle ((p@q) \cdot N) \rangle \prec ((p@q) \cdot P')$
apply $(\text{simp add: create-residual.simps})$
by (erule rev-mp) $(\text{subst boundOutputChainAlpha, auto})$

with $Sp Sq \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
have $\Psi \triangleright P \mapsto M(\nu*((q \cdot xvec)@(p \cdot yvec)))\langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
by(*simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm*)
moreover from $\langle x \in \text{supp } N \rangle$ **have** $((p@q) \cdot x) \in (p@q) \cdot (\text{supp } N)$
by(*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle Sp Sq$
have $x \in \text{supp}((p@q) \cdot N)$ **by**(*simp add: eqvts pt2[OF pt-name-inst]*)
moreover from $\langle \text{distinct}(xvec@yvec) \rangle$ **have** $\text{distinct}(q \cdot xvec)$ **and** $\text{distinct}(p \cdot yvec)$
by *auto*
moreover note $\langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle x \# M \rangle \langle x \# \Psi \rangle$
 $\langle (q \cdot xvec) \#* \Psi \rangle \langle (q \cdot xvec) \#* P \rangle \langle (q \cdot xvec) \#* M \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle$
 $\langle (p \cdot yvec) \#* \Psi \rangle \langle (p \cdot yvec) \#* P \rangle \langle (p \cdot yvec) \#* M \rangle \langle \text{distinct}(q \cdot xvec) \rangle$
ultimately have $\Psi \triangleright (\nu x)P \mapsto M(\nu*((q \cdot xvec)@x\#(p \cdot yvec)))\langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
by(*metis cOpen*)
with $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle$
 $\langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
 $Sp Sq$
have $\Psi \triangleright (\nu x)P \mapsto M(\nu*((p@q) \cdot (xvec@x\#yvec)))\langle((p@q) \cdot N)\rangle \prec ((p@q) \cdot P')$
by(*simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm*)
then show *?thesis* **using** $\langle((p@q) \cdot (xvec @ yvec)) \#* N \rangle \langle((p@q) \cdot (xvec @ yvec)) \#* P'\rangle Spq$
apply(*simp add: create-residual.simps*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha, auto*)
qed

lemma *BrOpen*:

fixes Ψ **::** $'b$
and P **::** $('a, 'b, 'c) psi$
and M **::** $'a$
and $xvec$ **::** *name list*
and $yvec$ **::** *name list*
and N **::** $'a$
and P' **::** $('a, 'b, 'c) psi$
and x **::** *name*

assumes *Trans*: $\Psi \triangleright P \mapsto_i M(\nu*(xvec@yvec))\langle N \rangle \prec P'$

and $x \in \text{supp } N$
and $x \# \Psi$
and $x \# M$
and $x \# xvec$
and $x \# yvec$

shows $\Psi \triangleright (\nu x)P \mapsto_i M(\nu*(xvec@x\#yvec))\langle N \rangle \prec P'$

proof –

from *Trans* **have** $\text{distinct}(xvec@yvec)$ **by**(*force dest: boundOutputDistinct*)

then have $xvec \#* yvec$ **by**(*induct xvec*) *auto*

obtain p **where** $(p \cdot yvec) \#* \Psi$ **and** $(p \cdot yvec) \#* P$ **and** $(p \cdot yvec) \#* M$
and $(p \cdot yvec) \#* yvec$ **and** $(p \cdot yvec) \#* N$ **and** $(p \cdot yvec) \#* P'$
and $x \# (p \cdot yvec)$ **and** $(p \cdot yvec) \#* xvec$
and $Sp: (set\ p) \subseteq (set\ yvec) \times (set(p \cdot yvec))$
by(*rule name-list-avoiding*[**where** $xvec=yvec$ **and** $c=(\Psi, P, M, xvec, yvec, N, P', x)$])
(auto simp add: eqvts fresh-star-prod)

obtain q **where** $(q \cdot xvec) \#* \Psi$ **and** $(q \cdot xvec) \#* P$ **and** $(q \cdot xvec) \#* M$
and $(q \cdot xvec) \#* xvec$ **and** $(q \cdot xvec) \#* N$ **and** $(q \cdot xvec) \#* P'$
and $x \# (q \cdot xvec)$ **and** $(q \cdot xvec) \#* yvec$
and $(q \cdot xvec) \#* p$ **and** $(q \cdot xvec) \#* (p \cdot yvec)$
and $Sq: (set\ q) \subseteq (set\ xvec) \times (set(q \cdot xvec))$
by(*rule name-list-avoiding*[**where** $xvec=xvec$ **and** $c=(\Psi, P, M, xvec, yvec, p \cdot yvec, N, P', x, p)$])
(auto simp add: eqvts fresh-star-prod)

note $\langle \Psi \triangleright P \mapsto_i M(\nu*(xvec@yvec)) \rangle(N) \prec P'$
moreover from $\langle (p \cdot yvec) \#* N \rangle \langle (q \cdot xvec) \#* N \rangle \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle Sp\ Sq$
have $((p@q) \cdot (xvec @ yvec)) \#* N$
apply(*simp only: eqvts*)
apply(*simp only: pt2[OF pt-name-inst]*)
by *simp*

moreover from $\langle (p \cdot yvec) \#* P' \rangle \langle (q \cdot xvec) \#* P' \rangle \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle Sp\ Sq$
have $((p@q) \cdot (xvec @ yvec)) \#* P'$ **by**(*simp del: freshAlphaPerm add: eqvts pt2[OF pt-name-inst]*)
moreover from $Sp\ Sq \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
have $Spq: set(p@q) \subseteq set(xvec@yvec) \times set((p@q) \cdot (xvec@yvec))$
by(*simp add: pt2[OF pt-name-inst] eqvts blast*)
ultimately have $\Psi \triangleright P \mapsto_i M(\nu*((p@q) \cdot (xvec@yvec))) \langle ((p@q) \cdot N) \rangle \prec ((p@q) \cdot P')$
apply(*simp add: create-residual.simps*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha, auto*)

with $Sp\ Sq \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
have $\Psi \triangleright P \mapsto_i M(\nu*((q \cdot xvec)@(p \cdot yvec))) \langle ((p@q) \cdot N) \rangle \prec ((p@q) \cdot P')$
by(*simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm*)
moreover from $\langle x \in supp\ N \rangle$ **have** $((p@q) \cdot x) \in (p@q) \cdot (supp\ N)$
by(*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle Sp\ Sq$
have $x \in supp((p@q) \cdot N)$ **by**(*simp add: eqvts pt2[OF pt-name-inst]*)
moreover from $\langle distinct(xvec@yvec) \rangle$ **have** $distinct(q \cdot xvec)$ **and** $distinct(p \cdot yvec)$
by *auto*

```

moreover note  $\langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle x \# M \rangle \langle x \# \Psi \rangle$ 
 $\langle (q \cdot xvec) \#* \Psi \rangle \langle (q \cdot xvec) \#* P \rangle \langle (q \cdot xvec) \#* M \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle$ 
 $\langle (p \cdot yvec) \#* \Psi \rangle \langle (p \cdot yvec) \#* P \rangle \langle (p \cdot yvec) \#* M \rangle \langle distinct(q \cdot xvec) \rangle$ 
ultimately have  $\Psi \triangleright (\nu x)P \mapsto_i M(\nu*((q \cdot xvec)@x\#(p \cdot yvec)))\langle (p@q) \cdot N \rangle$ 
 $\prec ((p@q) \cdot P')$ 
by(metis cBrOpen)
with  $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle$ 
 $\langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$ 
Sp Sq
have  $\Psi \triangleright (\nu x)P \mapsto_i M(\nu*((p@q) \cdot (xvec@x\#yvec)))\langle (p@q) \cdot N \rangle \prec ((p@q) \cdot$ 
 $P')$ 
by(simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm)
then show ?thesis using  $\langle (p@q) \cdot (xvec @ yvec) \rangle \#* N \rangle \langle (p@q) \cdot (xvec @$ 
 $yvec) \rangle \#* P' \rangle$  Spq
apply(simp add: create-residual.simps)
by(erule rev-mp) (subst boundOutputChainAlpha, auto)
qed

```

lemma *Scope:*

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\alpha$  :: 'a action
and  $P'$  :: ('a, 'b, 'c) psi
and  $x$  :: name

```

```

assumes  $\Psi \triangleright P \mapsto \alpha \prec P'$ 
and  $x \# \Psi$ 
and  $x \# \alpha$ 

```

shows $\Psi \triangleright (\nu x)P \mapsto \alpha \prec (\nu x)P'$

proof –

```

{
  fix  $\Psi P M xvec N P' x$ 

```

```

assume  $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $(x::name) \# \Psi$ 
and  $x \# M$ 
and  $x \# xvec$ 
and  $x \# N$ 

```

```

obtain  $p::name prm$  where  $(p \cdot xvec) \#* \Psi$  and  $(p \cdot xvec) \#* P$  and  $(p \cdot xvec)$ 
 $\#* M$  and  $(p \cdot xvec) \#* xvec$ 
and  $(p \cdot xvec) \#* N$  and  $(p \cdot xvec) \#* P'$  and  $x \# (p \cdot xvec)$ 
and  $S: (set p) \subseteq (set xvec) \times (set(p \cdot xvec))$ 
by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(\Psi, P, M, xvec, N, P',$ 
 $x)]$ )

```

(*auto simp add: eqvts fresh-star-prod*)

```

from  $\langle \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P' \rangle \langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle$   $S$ 
have  $\Psi \triangleright P \mapsto M(\nu*(p \cdot xvec))\langle (p \cdot N) \rangle \prec (p \cdot P')$ 

```



```

    by(simp add: boundOutputChainAlpha'' create-residual.simps)
  moreover then have distinct(p · xvec) by(force dest: boundOutputDistinct)
  moreover note ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # (p · xvec)⟩
  moreover from ⟨x # xvec⟩ ⟨x # p · xvec⟩ ⟨x # N⟩ S have x # (p · N)
    by(simp add: fresh-left del: fresh.AlphaSwap)
  ultimately have Ψ ▷ (νx)P ⟶M(ν*(p · xvec))⟨(p · N)⟩ < (νx)(p · P')
using ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec) #* P'⟩ ⟨(p · xvec) #* M⟩
  by(force intro: cScope)
  moreover from ⟨x # xvec⟩ ⟨x # p · xvec⟩ S have p · x = x by simp
  ultimately have Ψ ▷ (νx)P ⟶M(ν*(p · xvec))⟨(p · N)⟩ < (p · ((νx)P'))
by simp
  moreover from ⟨(p · xvec) #* P'⟩ ⟨x # xvec⟩ ⟨x # (p · xvec)⟩ have (p · xvec)
#* (νx)P'
    by(simp add: abs-fresh-star)
  ultimately have Ψ ▷ (νx)P ⟶M(ν*xvec)⟨N⟩ < (νx)P' using ⟨(p · xvec)
#* N⟩ S
    by(simp add: boundOutputChainAlpha'' create-residual.simps)
}
then have

  outputCase: ∧Ψ P M xvec N P' x.
  [[Ψ ▷ P ⟶M(ν*xvec)⟨N⟩ < P';
  (x::name) # Ψ;
  x # M;
  x # xvec;
  x # N]] ⟹
  Ψ ▷ (νx)P ⟶M(ν*xvec)⟨N⟩ < (νx)P' by simp

{
  fix Ψ P M xvec N P' x

  assume Ψ ▷ P ⟶M(ν*xvec)⟨N⟩ < P'
    and (x::name) # Ψ
    and x # M
    and x # xvec
    and x # N

  obtain p::name prm where (p · xvec) #* Ψ and (p · xvec) #* P and (p · xvec)
#* M and (p · xvec) #* xvec
    and (p · xvec) #* N and (p · xvec) #* P' and x # (p · xvec)
    and S: (set p) ⊆ (set xvec) × (set(p · xvec))
  by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, P, M, xvec, N, P',
x)])

  (auto simp add: eqvs fresh-star-prod)
  from ⟨Ψ ▷ P ⟶M(ν*xvec)⟨N⟩ < P'⟩ ⟨(p · xvec) #* N⟩ ⟨(p · xvec) #* P'⟩ S
  have Ψ ▷ P ⟶M(ν*(p · xvec))⟨(p · N)⟩ < (p · P')
    by(simp add: boundOutputChainAlpha'' create-residual.simps)
  moreover then have distinct(p · xvec) by(force dest: boundOutputDistinct)
  moreover note ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # (p · xvec)⟩

```

```

moreover from ⟨x # xvec⟩ ⟨x # p · xvec⟩ ⟨x # N⟩ S have x # (p · N)
  by(simp add: fresh-left del: freshAlphaSwap)
ultimately have Ψ ▷ (νx)P ↦iM(ν*(p · xvec))⟨(p · N)⟩ <(νx)⟨(p · P')⟩
using ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec) #* P'⟩ ⟨(p · xvec) #* M⟩
  by(force simp add: cScope)
moreover from ⟨x # xvec⟩ ⟨x # p · xvec⟩ S have p · x = x by simp
ultimately have Ψ ▷ (νx)P ↦iM(ν*(p · xvec))⟨(p · N)⟩ <(p · ((νx)P'))⟩
by simp
moreover from ⟨(p · xvec) #* P'⟩ ⟨x # xvec⟩ ⟨x # (p · xvec)⟩ have (p · xvec)
#* (νx)P'
  by(simp add: abs-fresh-star)
ultimately have Ψ ▷ (νx)P ↦iM(ν*xvec)⟨N⟩ <(νx)P' using ⟨(p · xvec)
#* N⟩ S
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
}
then have

```

```

  brouputCase: ∧Ψ P M xvec N P' x.
  [[Ψ ▷ P ↦iM(ν*xvec)⟨N⟩ <P'⟩;
  (x::name) # Ψ;
  x # M;
  x # xvec;
  x # N]] ⇒
  Ψ ▷ (νx)P ↦iM(ν*xvec)⟨N⟩ <(νx)P' by simp

```

```

show ?thesis
proof(induct rule: actionCases[where α=α])
  case(cInput M N)
    with assms show ?case by(force intro: cScope)
  next
    case(cBrInput M N)
      with assms show ?case by(force intro: cScope)
    next
      case(cOutput M xvec N)
        with assms show ?case by(force intro: outputCase)
      next
        case(cBrOutput M xvec N)
          with assms show ?case by(force intro: brouputCase)
        next
          case cTau
            with assms show ?case by(force intro: cScope)
        qed
      qed
    qed

```

```

lemma inputSwapFrameSubject:
fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi
  and M :: 'a
  and N :: 'a

```

```

and  $P' :: ('a, 'b, 'c) \text{ psi}$ 
and  $x :: \text{ name}$ 
and  $y :: \text{ name}$ 

assumes  $\Psi \triangleright P \mapsto M(N) \prec P'$ 
and  $x \# P$ 
and  $y \# P$ 

shows  $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([(x, y)] \cdot M)(N) \prec P'$ 
using assms
proof(nominal-induct avoiding: x y rule: inputInduct)
  case(cInput  $\Psi M K \text{ xvec } N \text{ Tvec } P x y$ )
    from  $\langle x \# M(\lambda * \text{xvec } N).P \rangle$  have  $x \# M$  by simp
    from  $\langle y \# M(\lambda * \text{xvec } N).P \rangle$  have  $y \# M$  by simp
    from  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot M) \leftrightarrow ([(x, y)] \cdot K)$ 
      by(rule chanEqClosed)
    with  $\langle x \# M \rangle \langle y \# M \rangle$  have  $([(x, y)] \cdot \Psi) \vdash M \leftrightarrow ([(x, y)] \cdot K)$ 
      by(simp)
    then show ?case using  $\langle \text{distinct xvec} \rangle \langle \text{set xvec} \subseteq \text{supp } N \rangle \langle \text{length xvec} = \text{length Tvec} \rangle$ 
      by(rule Input)
  next
    case(cCase  $\Psi P M N P' \varphi Cs x y$ )
      from  $\langle x \# \text{Cases } Cs \rangle \langle y \# \text{Cases } Cs \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$  have  $x \# \varphi$  and  $x \# P$ 
and  $y \# \varphi$  and  $y \# P$ 
      by(auto dest: memFresh)
      from  $\langle x \# P \rangle \langle y \# P \rangle$  have  $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([(x, y)] \cdot M)(N) \prec P'$  by(rule cCase)
      moreover note  $\langle (\varphi, P) \in \text{set } Cs \rangle$ 
      moreover from  $\langle \Psi \vdash \varphi \rangle$  have  $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$  by(rule statClosed)
      with  $\langle x \# \varphi \rangle \langle y \# \varphi \rangle$  have  $([(x, y)] \cdot \Psi) \vdash \varphi$  by simp
      ultimately show ?case using  $\langle \text{guarded } P \rangle$  by(rule Case)
  next
    case(cPar1  $\Psi \Psi_Q P M N P' A_Q Q x y$ )
      from  $\langle x \# P \parallel Q \rangle$  have  $x \# P$  and  $x \# Q$  by simp+
      from  $\langle y \# P \parallel Q \rangle$  have  $y \# P$  and  $y \# Q$  by simp+
      from  $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto ([(x, y)] \cdot M)(N) \prec P' \rangle$ 
      have  $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \triangleright P \mapsto ([(x, y)] \cdot M)(N) \prec P'$ 
      by(simp add: eqts)

      moreover from  $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$  have  $([(x, y)] \cdot (\text{extractFrame } Q)) = ([(x, y)] \cdot \langle A_Q, \Psi_Q \rangle)$ 
      by simp
      with  $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$  have  $\langle A_Q, ([(x, y)] \cdot \Psi_Q) \rangle = \text{extractFrame } Q$ 
      by(simp add: eqts)
      moreover from  $\langle A_Q \#* \Psi \rangle$  have  $([(x, y)] \cdot A_Q) \#* ([(x, y)] \cdot \Psi)$ 
      by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])

```

with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y] \cdot \Psi)$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $([x, y] \cdot A_Q) \#* ([x, y] \cdot M)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y] \cdot M)$ **by** *simp*
ultimately show *?case* **using** $\langle A_Q \#* P \rangle \langle A_Q \#* N \rangle$
by(*force intro!: Par1*)
next
case(*cPar2* $\Psi \Psi_P Q M N Q' A_P P x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*
from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ([x, y] \cdot (\Psi \otimes \Psi_P)) \triangleright Q$
 $\mapsto ([x, y] \cdot M)(N) \prec Q'$
have $([x, y] \cdot \Psi) \otimes ([x, y] \cdot \Psi_P) \triangleright Q \mapsto ([x, y] \cdot M)(N) \prec Q'$
by(*simp add: eqvts*)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $([x, y] \cdot (\text{extractFrame } P)) = ([x, y] \cdot \langle A_P, \Psi_P \rangle)$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\langle A_P, ([x, y] \cdot \Psi_P) \rangle = \text{extractFrame } P$
by(*simp add: eqvts*)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $([x, y] \cdot A_P) \#* ([x, y] \cdot \Psi)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y] \cdot \Psi)$ **by** *simp*
moreover from $\langle A_P \#* M \rangle$ **have** $([x, y] \cdot A_P) \#* ([x, y] \cdot M)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y] \cdot M)$ **by** *simp*
ultimately show *?case* **using** $\langle A_P \#* Q \rangle \langle A_P \#* N \rangle$
by(*force intro: Par2*)
next
case(*cScope* $\Psi P M N P' z x y$)
from $\langle x \# (\nu z)P \rangle \langle z \# x \rangle$ **have** $x \# P$ **by**(*simp add: abs-fresh*)
from $\langle y \# (\nu z)P \rangle \langle z \# y \rangle$ **have** $y \# P$ **by**(*simp add: abs-fresh*)
from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ([x, y] \cdot \Psi) \triangleright P \mapsto ([x, y] \cdot M)(N) \prec P'$
have $([x, y] \cdot \Psi) \triangleright P \mapsto ([x, y] \cdot M)(N) \prec P'$ **by** *simp*
moreover with $\langle z \# \Psi \rangle$ **have** $([x, y] \cdot z) \# ([x, y] \cdot \Psi)$
by(*simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle z \# x \rangle \langle z \# y \rangle$ **have** $z \# ([x, y] \cdot \Psi)$ **by** *simp*
moreover with $\langle z \# M \rangle$ **have** $([x, y] \cdot z) \# ([x, y] \cdot M)$
by(*simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle z \# x \rangle \langle z \# y \rangle$ **have** $z \# ([x, y] \cdot M)$ **by** *simp*
ultimately show *?case* **using** $\langle z \# N \rangle$
by(*force intro!: Scope*)
next
case(*cBang* $\Psi P M N P' x y$)
then show *?case* **by**(*force intro: Bang*)
qed

lemma *brinputSwapFrameSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)$ *psi*
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)$ *psi*
and $x :: name$
and $y :: name$

assumes $\Psi \triangleright P \mapsto_i M(N) \prec P'$

and $x \# P$
and $y \# P$

shows $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i ([(x, y)] \cdot M)(N) \prec P'$

using *assms*

proof(*nominal-induct avoiding: x y rule: brInputInduct*)

case(*cBrInput $\Psi K M xvec N Tvec P x y$*)

from $\langle x \# M(\lambda * xvec N).P \rangle$ **have** $x \# M$ **by** *simp*

from $\langle y \# M(\lambda * xvec N).P \rangle$ **have** $y \# M$ **by** *simp*

from $\langle \Psi \vdash K \succeq M \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot K) \succeq ([(x, y)] \cdot M)$

by(*rule chanInConClosed*)

with $\langle x \# M \rangle \langle y \# M \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot K) \succeq M$

by(*simp*)

then show *?case using $\langle distinct\ xvec \rangle \langle set\ xvec \subseteq\ supp\ N \rangle \langle length\ xvec = length\ Tvec \rangle$*

by(*rule BrInput*)

next

case(*cCase $\Psi P M N P' \varphi Cs x y$*)

from $\langle x \# Cases\ Cs \rangle \langle y \# Cases\ Cs \rangle \langle (\varphi, P) \in set\ Cs \rangle$ **have** $x \# \varphi$ **and** $x \# P$
and $y \# \varphi$ **and** $y \# P$

by(*auto dest: memFresh*)

from $\langle x \# P \rangle \langle y \# P \rangle$ **have** $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i ([(x, y)] \cdot M)(N) \prec P'$
by(*rule cCase*)

moreover note $\langle (\varphi, P) \in set\ Cs \rangle$

moreover from $\langle \Psi \vdash \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$ **by**(*rule statClosed*)

with $\langle x \# \varphi \rangle \langle y \# \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** *simp*

ultimately show *?case using $\langle guarded\ P \rangle$* **by**(*rule Case*)

next

case(*cPar1 $\Psi \Psi_Q P M N P' A_Q Q x y$*)

from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*

from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*

from $\langle x \# P \rangle \langle y \# P \rangle \langle \wedge x\ y. \llbracket x \# P; y \# P \rrbracket \implies ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i ([(x, y)] \cdot M)(N) \prec P' \rangle$

have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \triangleright P \mapsto_i ([(x, y)] \cdot M)(N) \prec P'$

by(*simp add: eqvts*)

moreover from $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $([(x, y)] \cdot (extractFrame\ Q)) = ([(x, y)] \cdot \langle A_Q, \Psi_Q \rangle)$

by *simp*

with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\langle A_Q, ((x, y)) \cdot \Psi_Q \rangle =$
extractFrame Q
by(*simp add: eqvts*)
moreover from $\langle A_Q \#* \Psi \rangle$ **have** $((x, y)) \cdot A_Q \#* ((x, y)) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y)) \cdot \Psi$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $((x, y)) \cdot A_Q \#* ((x, y)) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y)) \cdot M$ **by** *simp*
ultimately show *?case* **using** $\langle A_Q \#* P \rangle \langle A_Q \#* N \rangle$
by(*force intro!: Par1*)
next
case(*cPar2* $\Psi \Psi_P Q M N Q' A_P P x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*
from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ((x, y)) \cdot (\Psi \otimes \Psi_P) \triangleright Q$
 $\mapsto \iota((x, y)) \cdot M \langle N \rangle \prec Q'$
have $((x, y)) \cdot \Psi \otimes ((x, y)) \cdot \Psi_P \triangleright Q \mapsto \iota((x, y)) \cdot M \langle N \rangle \prec Q'$
by(*simp add: eqvts*)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $((x, y)) \cdot (\text{extractFrame } P) =$
 $((x, y)) \cdot \langle A_P, \Psi_P \rangle$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\langle A_P, ((x, y)) \cdot \Psi_P \rangle =$ *extract-*
Frame P
by(*simp add: eqvts*)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $((x, y)) \cdot A_P \#* ((x, y)) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ((x, y)) \cdot \Psi$ **by** *simp*
moreover from $\langle A_P \#* M \rangle$ **have** $((x, y)) \cdot A_P \#* ((x, y)) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ((x, y)) \cdot M$ **by** *simp*
ultimately show *?case* **using** $\langle A_P \#* Q \rangle \langle A_P \#* N \rangle$
by(*force intro!: Par2*)
next
case (*cBrMerge* $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*

from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $((x, y)) \cdot (\text{extractFrame } P) = ((x,$
 $y)) \cdot \langle A_P, \Psi_P \rangle$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\text{extractFrame } P = \langle A_P, ((x, y))$
 $\cdot \Psi_P \rangle$
by(*simp add: eqvts*)

from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $((x, y)) \cdot (\text{extractFrame } Q) = ((x,$
 $y)) \cdot \langle A_Q, \Psi_Q \rangle$
by *simp*

with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\text{extractFrame } Q = \langle A_Q, ([x, y]) \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)

from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ([x, y]) \cdot (\Psi \otimes \Psi_Q) \triangleright P \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec P'$
have $([x, y]) \cdot \Psi \otimes ([x, y]) \cdot \Psi_Q \triangleright P \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec P'$
by(*simp add: eqvts*)

moreover from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ([x, y]) \cdot (\Psi \otimes \Psi_P) \triangleright Q \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec Q'$
have $([x, y]) \cdot \Psi \otimes ([x, y]) \cdot \Psi_P \triangleright Q \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec Q'$
by(*simp add: eqvts*)

moreover note $\langle \text{extractFrame } P = \langle A_P, ([x, y]) \cdot \Psi_P \rangle \rangle \langle \text{extractFrame } Q = \langle A_Q, ([x, y]) \cdot \Psi_Q \rangle \rangle$

moreover from $\langle A_P \#* \Psi \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot \Psi$ **by** *simp*

moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot \Psi_Q$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot \Psi_Q$ **by** *simp*

moreover from $\langle A_P \#* M \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot M$ **by** *simp*

moreover from $\langle A_Q \#* \Psi \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot \Psi$ **by** *simp*

moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot \Psi_P$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot \Psi_P$ **by** *simp*

moreover from $\langle A_Q \#* M \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot M$ **by** *simp*

moreover note $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$
 $\langle A_P \#* P \rangle \langle A_P \#* N \rangle \langle A_P \#* P' \rangle \langle A_P \#* Q \rangle \langle A_P \#* Q' \rangle \langle A_P \#* A_Q \rangle$
 $\langle A_Q \#* P \rangle \langle A_Q \#* N \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* Q' \rangle$
ultimately show *?case*
by(*force intro!: semantics.cBrMerge*)

next

case(*cScope* $\Psi P M N P' z x y$)

from $\langle x \# (\nu z)P \rangle \langle z \# x \rangle$ **have** $x \# P$ **by**(*simp add: abs-fresh*)

from $\langle y \# (\nu z)P \rangle \langle z \# y \rangle$ **have** $y \# P$ **by**(*simp add: abs-fresh*)

from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ([x, y]) \cdot \Psi \triangleright P \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec P'$

have $([x, y]) \cdot \Psi \triangleright P \mapsto_i ([x, y]) \cdot M \langle N \rangle \prec P'$ **by** *simp*

moreover with $\langle z \# \Psi \rangle$ **have** $([x, y]) \cdot z \# ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst]*)

```

with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · Ψ by simp
moreover with ⟨z # M⟩ have [(x, y)] · z # [(x, y)] · M
  by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · M by simp
ultimately show ?case using ⟨z # N⟩
  by(force intro: Scope)
next
  case(cBang Ψ P M N P' x y)
  then show ?case by(force intro: Bang)
qed

lemma inputPermFrameSubject:
  fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi
    and M :: 'a
    and N :: 'a
    and P' :: ('a, 'b, 'c) psi
    and p :: name prm
    and Xs :: name set
    and Ys :: name set

assumes Ψ ▷ P ⟶ M(N) < P'
  and S: set p ⊆ Xs × Ys
  and Xs #* P
  and Ys #* P

shows (p · Ψ) ▷ P ⟶ (p · M)(N) < P'
  using S
proof(induct p)
  case Nil
  from ⟨Ψ ▷ P ⟶ M(N) < P'⟩
  show ?case by simp
next
  case(Cons a p)
  from ⟨set(a#p) ⊆ Xs × Ys⟩ have set p ⊆ Xs × Ys by auto
  with ⟨set p ⊆ Xs × Ys ⟹ (p · Ψ) ▷ P ⟶ (p · M)(N) < P'⟩
  have Trans: (p · Ψ) ▷ P ⟶ (p · M)(N) < P' by simp
  from ⟨set(a#p) ⊆ Xs × Ys⟩ show ?case
  proof(cases a)
    case (Pair x y)
    then have x ∈ Xs and y ∈ Ys
      using ⟨set(a#p) ⊆ Xs × Ys⟩ by auto
    with ⟨Xs #* P⟩ ⟨Ys #* P⟩ have x # P and y # P
      by(auto simp add: fresh-star-def)
    with Trans have [(x, y)] · p · Ψ ▷ P ⟶ [(x, y)] · p · M(N) < P'
      by(rule inputSwapFrameSubject)
    then show ?thesis
      using Pair by simp
  qed

```


qed

lemma *brinputPermFrameSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c) \text{ psi}$
and $p :: \text{name prm}$
and $Xs :: \text{name set}$
and $Ys :: \text{name set}$

assumes $\Psi \triangleright P \mapsto \iota M \langle N \rangle \prec P'$

and $S: \text{set } p \subseteq Xs \times Ys$
and $Xs \#* P$
and $Ys \#* P$

shows $(p \cdot \Psi) \triangleright P \mapsto \iota(p \cdot M) \langle N \rangle \prec P'$

using S

proof(*induct p*)

case *Nil*

from $\langle \Psi \triangleright P \mapsto \iota M \langle N \rangle \prec P' \rangle$

show *?case by simp*

next

case(*Cons a p*)

from $\langle \text{set}(a\#p) \subseteq Xs \times Ys \rangle$ have $\text{set } p \subseteq Xs \times Ys$ by *auto*

with $\langle \text{set } p \subseteq Xs \times Ys \implies (p \cdot \Psi) \triangleright P \mapsto \iota(p \cdot M) \langle N \rangle \prec P' \rangle$

have *Trans*: $(p \cdot \Psi) \triangleright P \mapsto \iota(p \cdot M) \langle N \rangle \prec P'$ by *simp*

from $\langle \text{set}(a\#p) \subseteq Xs \times Ys \rangle$ show *?case*

proof(*cases a*)

case (*Pair x y*)

then have $x \in Xs$ and $y \in Ys$

using $\langle \text{set}(a\#p) \subseteq Xs \times Ys \rangle$ by *auto*

with $\langle Xs \#* P \rangle \langle Ys \#* P \rangle$ have $x \# P$ and $y \# P$

by(*auto simp add: fresh-star-def*)

with *Trans* have $([(x, y)] \cdot p \cdot \Psi) \triangleright P \mapsto \iota([(x, y)] \cdot p \cdot M) \langle N \rangle \prec P'$

by(*rule brinputSwapFrameSubject*)

then show *?thesis*

using *Pair* by *simp*

qed

qed

lemma *inputSwapSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c) \text{ psi}$
and $x :: \text{name}$

and $y :: name$
assumes $\Psi \triangleright P \mapsto M(N) \prec P'$
and $x \# P$
and $y \# P$
and $x \# \Psi$
and $y \# \Psi$
shows $\Psi \triangleright P \mapsto ([x, y] \cdot M)(N) \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto M(N) \prec P' \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([x, y] \cdot \Psi) \triangleright P \mapsto ([x, y] \cdot M)(N) \prec P'$
by $(rule\ inputSwapFrameSubject)$
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **show** $?thesis$
by $simp$
qed

lemma *brinputSwapSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $y :: name$

assumes $\Psi \triangleright P \mapsto_i M(N) \prec P'$
and $x \# P$
and $y \# P$
and $x \# \Psi$
and $y \# \Psi$

shows $\Psi \triangleright P \mapsto_i ([x, y] \cdot M)(N) \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto_i M(N) \prec P' \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([x, y] \cdot \Psi) \triangleright P \mapsto_i ([x, y] \cdot M)(N) \prec P'$
by $(rule\ brinputSwapFrameSubject)$
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **show** $?thesis$
by $simp$
qed

lemma *inputPermSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $Xs :: name\ set$

and $Ys :: \text{name set}$
assumes $\Psi \triangleright P \mapsto M(\downarrow N) \prec P'$
and $S: \text{set } p \subseteq Xs \times Ys$
and $Xs \#* P$
and $Ys \#* P$
and $Xs \#* \Psi$
and $Ys \#* \Psi$
shows $\Psi \triangleright P \mapsto (p \cdot M)(\downarrow N) \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto M(\downarrow N) \prec P' \rangle S \langle Xs \#* P \rangle \langle Ys \#* P \rangle$
have $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\downarrow N) \prec P'$
by $(\text{rule } \text{inputPermFrameSubject})$
with $\langle Xs \#* \Psi \rangle \langle Ys \#* \Psi \rangle S$ **show** $?thesis$
by simp
qed

lemma $\text{brinputPermSubject}$:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{psi}$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c) \text{psi}$
and $p :: \text{name prm}$
and $Xs :: \text{name set}$
and $Ys :: \text{name set}$

assumes $\Psi \triangleright P \mapsto_i M(\downarrow N) \prec P'$
and $S: \text{set } p \subseteq Xs \times Ys$
and $Xs \#* P$
and $Ys \#* P$
and $Xs \#* \Psi$
and $Ys \#* \Psi$
shows $\Psi \triangleright P \mapsto_i (p \cdot M)(\downarrow N) \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto_i M(\downarrow N) \prec P' \rangle S \langle Xs \#* P \rangle \langle Ys \#* P \rangle$
have $(p \cdot \Psi) \triangleright P \mapsto_i (p \cdot M)(\downarrow N) \prec P'$
by $(\text{rule } \text{brinputPermFrameSubject})$
with $\langle Xs \#* \Psi \rangle \langle Ys \#* \Psi \rangle S$ **show** $?thesis$
by simp
qed

lemma inputSwapFrame :
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{psi}$
and $M :: 'a$
and $N :: 'a$

```

    and P' :: ('a, 'b, 'c) psi
    and x :: name
    and y :: name

assumes  $\Psi \triangleright P \mapsto M(N) \prec P'$ 
    and x # P
    and y # P
    and x # M
    and y # M

shows  $([(x, y)] \cdot \Psi) \triangleright P \mapsto M(N) \prec P'$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto M(N) \prec P' \rangle \langle x \# P \rangle \langle y \# P \rangle$ 
  have  $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([ (x, y) ] \cdot M)(N) \prec P'$ 
    by(rule inputSwapFrameSubject)
  with  $\langle x \# M \rangle \langle y \# M \rangle$  show ?thesis
    by simp
qed

lemma brinputSwapFrame:
  fixes  $\Psi :: 'b$ 
    and P :: ('a, 'b, 'c) psi
    and M :: 'a
    and N :: 'a
    and P' :: ('a, 'b, 'c) psi
    and x :: name
    and y :: name

assumes  $\Psi \triangleright P \mapsto_i M(N) \prec P'$ 
    and x # P
    and y # P
    and x # M
    and y # M

shows  $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i M(N) \prec P'$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto_i M(N) \prec P' \rangle \langle x \# P \rangle \langle y \# P \rangle$ 
  have  $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i ([ (x, y) ] \cdot M)(N) \prec P'$ 
    by(rule brinputSwapFrameSubject)
  with  $\langle x \# M \rangle \langle y \# M \rangle$  show ?thesis
    by simp
qed

lemma inputPermFrame:
  fixes  $\Psi :: 'b$ 
    and P :: ('a, 'b, 'c) psi
    and M :: 'a
    and N :: 'a
    and P' :: ('a, 'b, 'c) psi

```

```

and  $p :: \text{name prm}$ 
and  $Xs :: \text{name set}$ 
and  $Ys :: \text{name set}$ 

assumes  $\Psi \triangleright P \mapsto M(N) \prec P'$ 
and  $S: \text{set } p \subseteq Xs \times Ys$ 
and  $Xs \#* P$ 
and  $Ys \#* P$ 
and  $Xs \#* M$ 
and  $Ys \#* M$ 

shows  $(p \cdot \Psi) \triangleright P \mapsto M(N) \prec P'$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto M(N) \prec P' \rangle S \langle Xs \#* P \rangle \langle Ys \#* P \rangle$ 
  have  $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(N) \prec P'$ 
    by(rule inputPermFrameSubject)
  with  $\langle Xs \#* M \rangle \langle Ys \#* M \rangle S$  show ?thesis
    by simp
qed

```

lemma *brinputPermFrame*:

```

fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) \text{psi}$ 
and  $M :: 'a$ 
and  $N :: 'a$ 
and  $P' :: ('a, 'b, 'c) \text{psi}$ 
and  $p :: \text{name prm}$ 
and  $Xs :: \text{name set}$ 
and  $Ys :: \text{name set}$ 

```

```

assumes  $\Psi \triangleright P \mapsto_i M(N) \prec P'$ 
and  $S: \text{set } p \subseteq Xs \times Ys$ 
and  $Xs \#* P$ 
and  $Ys \#* P$ 
and  $Xs \#* M$ 
and  $Ys \#* M$ 

shows  $(p \cdot \Psi) \triangleright P \mapsto_i M(N) \prec P'$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto_i M(N) \prec P' \rangle S \langle Xs \#* P \rangle \langle Ys \#* P \rangle$ 
  have  $(p \cdot \Psi) \triangleright P \mapsto_i (p \cdot M)(N) \prec P'$ 
    by(rule brinputPermFrameSubject)
  with  $\langle Xs \#* M \rangle \langle Ys \#* M \rangle S$  show ?thesis
    by simp
qed

```

lemma *inputAlpha*:

```

fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) \text{psi}$ 

```

```

and  $M$   :: 'a
and  $N$   :: 'a
and  $P'$  :: ('a, 'b, 'c) psi
and  $p$   :: name prm
and  $xvec$  :: name list

assumes  $\Psi \triangleright P \mapsto M(\downarrow N) \prec P'$ 
and  $set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec))$ 
and  $distinctPerm\ p$ 
and  $xvec \#* P$ 
and  $(p \cdot xvec) \#* P$ 

shows  $\Psi \triangleright P \mapsto M(\downarrow (p \cdot N)) \prec (p \cdot P')$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto M(\downarrow N) \prec P' \rangle \langle set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec)) \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle$ 
  have  $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\downarrow N) \prec P'$  by - (rule inputPermFrameSubject, auto)
  then have  $(p \cdot p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot ((p \cdot M)(\downarrow N) \prec P'))$  by(rule eqvts)
  with  $\langle distinctPerm\ p \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle \langle set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec)) \rangle$ 
  show ?thesis by(simp add: eqvts)
qed

lemma brinputAlpha:
  fixes  $\Psi$   :: 'b
  and  $P$   :: ('a, 'b, 'c) psi
  and  $M$   :: 'a
  and  $N$   :: 'a
  and  $P'$  :: ('a, 'b, 'c) psi
  and  $p$   :: name prm
  and  $xvec$  :: name list

assumes  $\Psi \triangleright P \mapsto_i M(\downarrow N) \prec P'$ 
and  $set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec))$ 
and  $distinctPerm\ p$ 
and  $xvec \#* P$ 
and  $(p \cdot xvec) \#* P$ 

shows  $\Psi \triangleright P \mapsto_i M(\downarrow (p \cdot N)) \prec (p \cdot P')$ 
proof -
  from  $\langle \Psi \triangleright P \mapsto_i M(\downarrow N) \prec P' \rangle \langle set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec)) \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle$ 
  have  $(p \cdot \Psi) \triangleright P \mapsto_i (p \cdot M)(\downarrow N) \prec P'$  by - (rule brinputPermFrameSubject, auto)
  then have  $(p \cdot p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot (i(p \cdot M)(\downarrow N) \prec P'))$  by(rule eqvts)
  with  $\langle distinctPerm\ p \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle \langle set\ p \subseteq (set\ xvec) \times (set\ (p \cdot xvec)) \rangle$ 
  show ?thesis by(simp add: eqvts)

```

qed

lemma *frameFresh*[*dest*]:

fixes $x :: \text{name}$
and $A_F :: \text{name list}$
and $\Psi_F :: 'b$

assumes $x \# A_F$
and $x \# \langle A_F, \Psi_F \rangle$

shows $x \# \Psi_F$
using *assms*
by(*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)

lemma *outputSwapFrameSubject*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $xvec :: \text{name list}$
and $N :: 'a$
and $x :: \text{name}$
and $y :: \text{name}$

assumes $\Psi \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'$
and $xvec \#^* M$
and $x \# P$
and $y \# P$

shows $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
using *assms*

proof(*nominal-induct avoiding: x y rule: outputInduct'*)

case *cAlpha*

then show ?*case* **by**(*simp add: create-residual.simps boundOutputChainAlpha''*)

next

case(*cOutput* $\Psi M K N P x y$)

from $\langle x \# M\langle N \rangle.P \rangle$ **have** $x \# M$ **by** *simp*

from $\langle y \# M\langle N \rangle.P \rangle$ **have** $y \# M$ **by** *simp*

from $\langle \Psi \vdash M \leftrightarrow K \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot M) \leftrightarrow ([(x, y)] \cdot K)$
by(*rule chanEqClosed*)

with $\langle x \# M \rangle \langle y \# M \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash M \leftrightarrow ([(x, y)] \cdot K)$

by(*simp*)

then show ?*case* **by**(*rule Output*)

next

case(*cCase* $\Psi P M xvec N P' \varphi Cs x y$)

from $\langle x \# \text{Cases } Cs \rangle \langle y \# \text{Cases } Cs \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$ **have** $x \# \varphi$ **and** $x \# P$
and $y \# \varphi$ **and** $y \# P$

by(*auto dest: memFresh*)

from $\langle x \# P \rangle \langle y \# P \rangle$ **have** $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$ **by**(*rule cCase*)

moreover note $\langle \varphi, P \rangle \in \text{set } Cs$
moreover from $\langle \Psi \vdash \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ([(x, y)] \cdot \varphi)$ **by** (rule statClosed)
with $\langle x \# \varphi \rangle \langle y \# \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** simp
ultimately show $?case$ **using** $\langle \text{guarded } P \rangle$ **by** (rule Case)
next
case $(cPar1 \Psi \Psi_Q P M xvec N P' A_Q Q x y)$
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** simp+
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** simp+
from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \triangleright P$
 $\mapsto ([(x, y)] \cdot M)(\nu * xvec) \langle N \rangle \prec P'$
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q) \triangleright P \mapsto ([(x, y)] \cdot M)(\nu * xvec) \langle N \rangle \prec P'$
by (simp add: eqvts)

moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_Q, \Psi_Q \rangle) =$
 $([(x, y)] \cdot (\text{extractFrame } Q))$
by simp
with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\langle A_Q, ([(x, y)] \cdot \Psi_Q) \rangle =$
 $\text{extractFrame } Q$
by (simp add: eqvts)
moreover from $\langle A_Q \#* \Psi \rangle$ **have** $([(x, y)] \cdot A_Q) \#* ([(x, y)] \cdot \Psi)$
by $(\text{simp add: pt-fresh-star-bij}[OF \text{ pt-name-inst}, OF \text{ at-name-inst}])$
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([(x, y)] \cdot \Psi)$ **by** simp
moreover from $\langle A_Q \#* M \rangle$ **have** $([(x, y)] \cdot A_Q) \#* ([(x, y)] \cdot M)$
by $(\text{simp add: pt-fresh-star-bij}[OF \text{ pt-name-inst}, OF \text{ at-name-inst}])$
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([(x, y)] \cdot M)$ **by** simp
ultimately show $?case$ **using** $\langle A_Q \#* P \rangle \langle A_Q \#* N \rangle \langle xvec \#* Q \rangle \langle A_Q \#* xvec \rangle$
by $(\text{force intro: Par1})$
next
case $(cPar2 \Psi \Psi_P Q M xvec N Q' A_P P x y)$
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** simp+
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** simp+
from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ([(x, y)] \cdot (\Psi \otimes \Psi_P)) \triangleright Q$
 $\mapsto ([(x, y)] \cdot M)(\nu * xvec) \langle N \rangle \prec Q'$
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \triangleright Q \mapsto ([(x, y)] \cdot M)(\nu * xvec) \langle N \rangle \prec Q'$
by (simp add: eqvts)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_P, \Psi_P \rangle) =$
 $([(x, y)] \cdot (\text{extractFrame } P))$
by simp
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\langle A_P, ([(x, y)] \cdot \Psi_P) \rangle = \text{extract-}$
 $\text{Frame } P$
by (simp add: eqvts)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot \Psi)$
by $(\text{simp add: pt-fresh-star-bij}[OF \text{ pt-name-inst}, OF \text{ at-name-inst}])$
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([(x, y)] \cdot \Psi)$ **by** simp
moreover from $\langle A_P \#* M \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot M)$
by $(\text{simp add: pt-fresh-star-bij}[OF \text{ pt-name-inst}, OF \text{ at-name-inst}])$
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([(x, y)] \cdot M)$ **by** simp
ultimately show $?case$ **using** $\langle A_P \#* Q \rangle \langle A_P \#* N \rangle \langle xvec \#* P \rangle \langle A_P \#* xvec \rangle$


```

  by(force intro: Par2)
next
  case(cOpen Ψ P M xvec yvec N P' z x y)
  from ⟨x # (νz)P⟩ ⟨z # x⟩ have x # P by(simp add: abs-fresh)
  from ⟨y # (νz)P⟩ ⟨z # y⟩ have y # P by(simp add: abs-fresh)
  from ⟨x # P⟩ ⟨y # P⟩ ⟨∧x y. [x # P; y # P] ⟹ [(x, y)] · Ψ ▷ P ⟶ [(x, y)] ·
M)(ν*(xvec@yvec))⟨N⟩ < P'⟩
  have [(x, y)] · Ψ ▷ P ⟶ [(x, y)] · M)(ν*(xvec@yvec))⟨N⟩ < P' by simp
  moreover with ⟨z # Ψ⟩ have [(x, y)] · z # [(x, y)] · Ψ
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · Ψ by simp
  moreover with ⟨z # M⟩ have [(x, y)] · z # [(x, y)] · M
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · M by simp
  ultimately show ?case using ⟨z ∈ supp N⟩ ⟨z # xvec⟩ ⟨z # yvec⟩
    by(force intro!: Open)
next
  case(cScope Ψ P M xvec N P' z x y)
  from ⟨x # (νz)P⟩ ⟨z # x⟩ have x # P by(simp add: abs-fresh)
  from ⟨y # (νz)P⟩ ⟨z # y⟩ have y # P by(simp add: abs-fresh)
  from ⟨x # P⟩ ⟨y # P⟩ ⟨∧x y. [x # P; y # P] ⟹ [(x, y)] · Ψ ▷ P ⟶ [(x, y)] ·
M)(ν*xvec)⟨N⟩ < P'⟩
  have [(x, y)] · Ψ ▷ P ⟶ [(x, y)] · M)(ν*xvec)⟨N⟩ < P' by simp
  moreover with ⟨z # Ψ⟩ have [(x, y)] · z # [(x, y)] · Ψ
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · Ψ by simp
  moreover with ⟨z # M⟩ have [(x, y)] · z # [(x, y)] · M
    by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · M by simp
  ultimately show ?case using ⟨z # N⟩ ⟨z # xvec⟩
    by(force intro!: Scope)
next
  case(cBang Ψ P M B x y)
  then show ?case by(force intro: Bang)
qed

lemma broutputSwapFrameSubject:
  fixes Ψ    :: 'b
  and P      :: ('a, 'b, 'c) psi
  and M      :: 'a
  and xvec   :: name list
  and N      :: 'a
  and x      :: name
  and y      :: name

assumes Ψ ▷ P ⟶i M(ν*xvec)⟨N⟩ < P'
  and xvec #* M
  and x # P
  and y # P

```

shows $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
using *assms*
proof(*nominal-induct avoiding: x y rule: brOutputInduct'*)
case *cAlpha*
then show *?case* **by**(*simp add: create-residual.simps boundOutputChainAlpha''*)
next
case(*cBrOutput* $\Psi M K N P x y$)
from $\langle x \# M \langle N \rangle . P \rangle$ **have** $x \# M$ **by** *simp*
from $\langle y \# M \langle N \rangle . P \rangle$ **have** $y \# M$ **by** *simp*
from $\langle \Psi \vdash M \preceq K \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ((x, y)] \cdot M) \preceq ((x, y)] \cdot K)$
by(*rule chanOutConClosed*)
with $\langle x \# M \rangle \langle y \# M \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash M \preceq ((x, y)] \cdot K)$
by(*simp*)
then show *?case* **by**(*rule BrOutput*)
next
case(*cCase* $\Psi P M xvec N P' \varphi Cs x y$)
from $\langle x \# Cases Cs \rangle \langle y \# Cases Cs \rangle \langle (\varphi, P) \in set Cs \rangle$ **have** $x \# \varphi$ **and** $x \# P$
and $y \# \varphi$ **and** $y \# P$
by(*auto dest: memFresh*)
from $\langle x \# P \rangle \langle y \# P \rangle$ **have** $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$ **by**(*rule cCase*)
moreover note $\langle (\varphi, P) \in set Cs \rangle$
moreover from $\langle \Psi \vdash \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash ((x, y)] \cdot \varphi)$ **by**(*rule statClosed*)
with $\langle x \# \varphi \rangle \langle y \# \varphi \rangle$ **have** $([(x, y)] \cdot \Psi) \vdash \varphi$ **by** *simp*
ultimately show *?case* **using** $\langle guarded P \rangle$ **by**(*rule Case*)
next
case(*cPar1* $\Psi \Psi_Q P M xvec N P' A_Q Q x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*
from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ((x, y)] \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
have $([(x, y)] \cdot \Psi) \otimes ((x, y)] \cdot \Psi_Q) \triangleright P \mapsto_i([(x, y)] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
by(*simp add: eqvts*)

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_Q, \Psi_Q \rangle) = ((x, y)] \cdot (extractFrame Q))$
by *simp*
with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\langle A_Q, ((x, y)] \cdot \Psi_Q) \rangle = extractFrame Q$
by(*simp add: eqvts*)
moreover from $\langle A_Q \#* \Psi \rangle$ **have** $([(x, y)] \cdot A_Q) \#* ((x, y)] \cdot \Psi)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y)] \cdot \Psi)$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $([(x, y)] \cdot A_Q) \#* ((x, y)] \cdot M)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** $\langle A_Q \#* P \rangle \langle A_Q \#* N \rangle \langle xvec \#* Q \rangle \langle A_Q \#* xvec \rangle$
by(*force intro: Par1*)

next
case(*cPar2* $\Psi \Psi_P Q M \nu xvec N Q' A_P P x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*
from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ((x, y) \cdot (\Psi \otimes \Psi_P)) \triangleright Q$
 $\mapsto_i([(x, y)] \cdot M)(\nu xvec)\langle N \rangle \prec Q'$
have $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \triangleright Q \mapsto_i([(x, y)] \cdot M)(\nu xvec)\langle N \rangle \prec Q'$
by(*simp add: eqvts*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_P, \Psi_P \rangle) =$
 $([(x, y)] \cdot (extractFrame P))$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\langle A_P, ([(x, y)] \cdot \Psi_P) \rangle = extract-$
Frame P
by(*simp add: eqvts*)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot \Psi)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([(x, y)] \cdot \Psi)$ **by** *simp*
moreover from $\langle A_P \#* M \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot M)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([(x, y)] \cdot M)$ **by** *simp*
ultimately show *?case* **using** $\langle A_P \#* Q \rangle \langle A_P \#* N \rangle \langle xvec \#* P \rangle \langle A_P \#* xvec \rangle$
by(*force intro: Par2*)

next
case(*cBrComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M \langle N \rangle \prec P' \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([(x, y)] \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i([(x, y)] \cdot M) \langle N \rangle \prec P'$
by(*rule brinputSwapFrameSubject*)
then have *permIn*: $(([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_Q)) \triangleright P \mapsto_i([(x, y)] \cdot M) \langle N \rangle$
 $\prec P'$
by(*simp add: eqvts*)
moreover from $\langle x \# Q \rangle \langle y \# Q \rangle \langle \bigwedge x y. \llbracket x \# Q; y \# Q \rrbracket \implies ((x, y) \cdot (\Psi \otimes \Psi_P))$
 $\triangleright Q \mapsto_i([(x, y)] \cdot M)(\nu xvec)\langle N \rangle \prec Q'$
have *permOut*: $([(x, y)] \cdot \Psi) \otimes ([(x, y)] \cdot \Psi_P) \triangleright Q \mapsto_i([(x, y)] \cdot M)(\nu xvec)\langle N \rangle$
 $\prec Q'$
by(*simp add: eqvts*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_P, \Psi_P \rangle) =$
 $([(x, y)] \cdot (extractFrame P))$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $extractFrame P = \langle A_P, ([(x, y)] \cdot \Psi_P) \rangle$
by(*simp add: eqvts*)
moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $([(x, y)] \cdot \langle A_Q, \Psi_Q \rangle) =$
 $([(x, y)] \cdot (extractFrame Q))$
by *simp*

with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\text{extractFrame } Q = \langle A_Q, ([x, y]) \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot \Psi$ **by** *simp*
moreover from $\langle A_Q \#* \Psi \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot \Psi$ **by** *simp*
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot \Psi_Q$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot \Psi_Q$ **by** *simp*
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot \Psi_P$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot \Psi_P$ **by** *simp*
moreover from $\langle A_P \#* M \rangle$ **have** $([x, y]) \cdot A_P \#* ([x, y]) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ([x, y]) \cdot M$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $([x, y]) \cdot A_Q \#* ([x, y]) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ([x, y]) \cdot M$ **by** *simp*
moreover from $\langle \text{vec} \#* \Psi \rangle$ **have** $([x, y]) \cdot \text{vec} \#* ([x, y]) \cdot \Psi$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{vec} \#* x \rangle \langle \text{vec} \#* y \rangle$ **have** $\text{vec} \#* ([x, y]) \cdot \Psi$ **by** *simp*
moreover from $\langle \text{vec} \#* \Psi_Q \rangle$ **have** $([x, y]) \cdot \text{vec} \#* ([x, y]) \cdot \Psi_Q$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{vec} \#* x \rangle \langle \text{vec} \#* y \rangle$ **have** $\text{vec} \#* ([x, y]) \cdot \Psi_Q$ **by** *simp*
moreover from $\langle \text{vec} \#* \Psi_P \rangle$ **have** $([x, y]) \cdot \text{vec} \#* ([x, y]) \cdot \Psi_P$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{vec} \#* x \rangle \langle \text{vec} \#* y \rangle$ **have** $\text{vec} \#* ([x, y]) \cdot \Psi_P$ **by** *simp*
moreover from $\langle \text{vec} \#* M \rangle$ **have** $([x, y]) \cdot \text{vec} \#* ([x, y]) \cdot M$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{vec} \#* x \rangle \langle \text{vec} \#* y \rangle$ **have** $\text{vec} \#* ([x, y]) \cdot M$ **by** *simp*

moreover note $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* N \rangle \langle A_P \#* P' \rangle$
 $\langle A_P \#* Q \rangle \langle A_P \#* Q' \rangle \langle A_P \#* A_Q \rangle$
 $\langle A_P \#* \text{vec} \rangle \langle A_Q \#* P \rangle \langle A_Q \#* N \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* Q' \rangle \langle A_Q \#*$
 $\text{vec} \rangle$
 $\langle \text{distinct } \text{vec} \rangle \langle \text{vec} \#* P \rangle \langle \text{vec} \#* Q \rangle$

ultimately show *?case*
by(*simp add: semantics.cBrComm1*)

next
case(*cBrComm2* $\Psi \Psi_Q P M \text{vec } N P' A_P \Psi_P Q Q' A_Q x y$)
from $\langle x \# P \parallel Q \rangle$ **have** $x \# P$ **and** $x \# Q$ **by** *simp+*
from $\langle y \# P \parallel Q \rangle$ **have** $y \# P$ **and** $y \# Q$ **by** *simp+*

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{!}M(\text{!}N) \prec Q' \rangle \langle x \# Q \rangle \langle y \# Q \rangle$
have $([x, y]) \cdot (\Psi \otimes \Psi_P) \triangleright Q \mapsto \text{!}([x, y]) \cdot M(\text{!}N) \prec Q'$
by(*rule brinputSwapFrameSubject*)

then have $\text{permIn}: (((x, y) \cdot \Psi) \otimes ((x, y) \cdot \Psi_P)) \triangleright Q \mapsto_i ((x, y) \cdot M) \langle N \rangle$
 $\prec Q'$
by (*simp add: eqvts*)
moreover from $\langle x \# P \rangle \langle y \# P \rangle \langle \bigwedge x y. \llbracket x \# P; y \# P \rrbracket \implies ((x, y) \cdot (\Psi \otimes \Psi_Q))$
 $\triangleright P \mapsto_i (((x, y) \cdot M) \langle \nu * \text{xvec} \rangle \langle N \rangle) \prec P'$
have $\text{permOut}: (((x, y) \cdot \Psi) \otimes ((x, y) \cdot \Psi_Q)) \triangleright P \mapsto_i ((x, y) \cdot M) \langle \nu * \text{xvec} \rangle \langle N \rangle$
 $\prec P'$
by (*simp add: eqvts*)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $((x, y) \cdot \langle A_P, \Psi_P \rangle) =$
 $((x, y) \cdot (\text{extractFrame } P))$
by *simp*
with $\langle A_P \#* x \rangle \langle x \# P \rangle \langle A_P \#* y \rangle \langle y \# P \rangle$ **have** $\text{extractFrame } P = \langle A_P, ((x, y) \cdot \Psi_P) \rangle$
by (*simp add: eqvts*)
moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $((x, y) \cdot \langle A_Q, \Psi_Q \rangle) =$
 $((x, y) \cdot (\text{extractFrame } Q))$
by *simp*
with $\langle A_Q \#* x \rangle \langle x \# Q \rangle \langle A_Q \#* y \rangle \langle y \# Q \rangle$ **have** $\text{extractFrame } Q = \langle A_Q, ((x, y) \cdot \Psi_Q) \rangle$
by (*simp add: eqvts*)
moreover from $\langle A_P \#* \Psi \rangle$ **have** $((x, y) \cdot A_P) \#* ((x, y) \cdot \Psi)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ((x, y) \cdot \Psi)$ **by** *simp*
moreover from $\langle A_Q \#* \Psi \rangle$ **have** $((x, y) \cdot A_Q) \#* ((x, y) \cdot \Psi)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y) \cdot \Psi)$ **by** *simp*
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $((x, y) \cdot A_P) \#* ((x, y) \cdot \Psi_Q)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ((x, y) \cdot \Psi_Q)$ **by** *simp*
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $((x, y) \cdot A_Q) \#* ((x, y) \cdot \Psi_P)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y) \cdot \Psi_P)$ **by** *simp*
moreover from $\langle A_P \#* M \rangle$ **have** $((x, y) \cdot A_P) \#* ((x, y) \cdot M)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle A_P \#* y \rangle$ **have** $A_P \#* ((x, y) \cdot M)$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $((x, y) \cdot A_Q) \#* ((x, y) \cdot M)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_Q \#* x \rangle \langle A_Q \#* y \rangle$ **have** $A_Q \#* ((x, y) \cdot M)$ **by** *simp*
moreover from $\langle \text{xvec} \#* \Psi \rangle$ **have** $((x, y) \cdot \text{xvec}) \#* ((x, y) \cdot \Psi)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{xvec} \#* x \rangle \langle \text{xvec} \#* y \rangle$ **have** $\text{xvec} \#* ((x, y) \cdot \Psi)$ **by** *simp*
moreover from $\langle \text{xvec} \#* \Psi_Q \rangle$ **have** $((x, y) \cdot \text{xvec}) \#* ((x, y) \cdot \Psi_Q)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{xvec} \#* x \rangle \langle \text{xvec} \#* y \rangle$ **have** $\text{xvec} \#* ((x, y) \cdot \Psi_Q)$ **by** *simp*
moreover from $\langle \text{xvec} \#* \Psi_P \rangle$ **have** $((x, y) \cdot \text{xvec}) \#* ((x, y) \cdot \Psi_P)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{xvec} \#* x \rangle \langle \text{xvec} \#* y \rangle$ **have** $\text{xvec} \#* ((x, y) \cdot \Psi_P)$ **by** *simp*
moreover from $\langle \text{xvec} \#* M \rangle$ **have** $((x, y) \cdot \text{xvec}) \#* ((x, y) \cdot M)$

```

  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  with ⟨xvec #* x⟩ ⟨xvec #* y⟩ have xvec #* [(x, y)] · M by simp

  moreover note ⟨distinct AP⟩ ⟨distinct AQ⟩ ⟨AP #* P⟩ ⟨AP #* N⟩ ⟨AP #* P'⟩
  ⟨AP #* Q⟩ ⟨AP #* Q'⟩ ⟨AP #* AQ⟩
  ⟨AP #* xvec⟩ ⟨AQ #* P⟩ ⟨AQ #* N⟩ ⟨AQ #* P'⟩ ⟨AQ #* Q⟩ ⟨AQ #* Q'⟩ ⟨AQ #*
  xvec⟩
  ⟨distinct xvec⟩ ⟨xvec #* P⟩ ⟨xvec #* Q⟩
  ultimately show ?case
  by(simp add: semantics.cBrComm2)
next
case(cBrOpen Ψ P M xvec yvec N P' z x y)
from ⟨x # (νz)P⟩ ⟨z # x⟩ have x # P by(simp add: abs-fresh)
from ⟨y # (νz)P⟩ ⟨z # y⟩ have y # P by(simp add: abs-fresh)
from ⟨x # P⟩ ⟨y # P⟩ ⟨∧x y. [x # P; y # P] ⇒ [(x, y)] · Ψ⟩ ▷ P ↦i([(x, y)]
· M)(ν*(xvec@yvec))(N) < P'
have [(x, y)] · Ψ ▷ P ↦i([(x, y)] · M)(ν*(xvec@yvec))(N) < P' by simp
moreover with ⟨z # Ψ⟩ have [(x, y)] · z # [(x, y)] · Ψ
  by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · Ψ by simp
moreover with ⟨z # M⟩ have [(x, y)] · z # [(x, y)] · M
  by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · M by simp
ultimately show ?case using ⟨z ∈ supp N⟩ ⟨z # xvec⟩ ⟨z # yvec⟩
  by(force intro: BrOpen)
next
case(cScope Ψ P M xvec N P' z x y)
from ⟨x # (νz)P⟩ ⟨z # x⟩ have x # P by(simp add: abs-fresh)
from ⟨y # (νz)P⟩ ⟨z # y⟩ have y # P by(simp add: abs-fresh)
from ⟨x # P⟩ ⟨y # P⟩ ⟨∧x y. [x # P; y # P] ⇒ [(x, y)] · Ψ⟩ ▷ P ↦i([(x, y)]
· M)(ν*xvec)(N) < P'
have [(x, y)] · Ψ ▷ P ↦i([(x, y)] · M)(ν*xvec)(N) < P' by simp
moreover with ⟨z # Ψ⟩ have [(x, y)] · z # [(x, y)] · Ψ
  by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · Ψ by simp
moreover with ⟨z # M⟩ have [(x, y)] · z # [(x, y)] · M
  by(simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst])
with ⟨z # x⟩ ⟨z # y⟩ have z # [(x, y)] · M by simp
ultimately show ?case using ⟨z # N⟩ ⟨z # xvec⟩
  by(force intro: Scope)
next
case(cBang Ψ P M B x y)
then show ?case by(force intro: Bang)
qed

lemma outputPermFrameSubject:
  fixes Ψ      :: 'b
  and P       :: ('a, 'b, 'c) psi
  and M       :: 'a

```

```

and  $xvec$  :: name list
and  $N$  :: 'a
and  $P'$  :: ('a, 'b, 'c) psi
and  $p$  :: name prm
and  $yvec$  :: name list
and  $zvec$  :: name list

assumes  $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$ 
and  $S$ : set  $p \subseteq$  set  $yvec \times$  set  $zvec$ 
and  $yvec \#* P$ 
and  $zvec \#* P$ 

shows  $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\nu*xvec)\langle N \rangle \prec P'$ 
proof -
  {
    fix  $xvec N P' Xs YS$ 
    assume  $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$  and  $xvec \#* M$  and  $xvec \#* yvec$  and
 $xvec \#* zvec$ 
    have  $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\nu*xvec)\langle N \rangle \prec P'$  using  $S$ 
    proof(induct p)
      case Nil
      from  $\langle \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P' \rangle$ 
      show ?case by simp
    next
      case(Cons a p)
      from  $\langle set(a\#p) \subseteq set yvec \times set zvec \rangle$  have set  $p \subseteq$  set  $yvec \times$  set  $zvec$  by
auto
      then have Trans:  $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\nu*xvec)\langle N \rangle \prec P'$  by(rule Cons)
      show ?case
      proof(cases a)
        case (Pair x y)
        note Trans
        moreover from  $\langle xvec \#* yvec \rangle \langle xvec \#* zvec \rangle \langle set p \subseteq set yvec \times set zvec \rangle$ 
 $\langle xvec \#* M \rangle$  have  $xvec \#* (p \cdot M)$ 
          by(simp add: freshChainSimps)
        moreover have  $x \in set yvec$  and  $y \in set zvec$ 
          using  $\langle set(a\#p) \subseteq set yvec \times set zvec \rangle$  Pair by auto
        with  $\langle yvec \#* P \rangle \langle zvec \#* P \rangle$  have  $x \# P$  and  $y \# P$ 
          by(auto simp add: fresh-star-def)
        ultimately have  $([(x, y)] \cdot p \cdot \Psi) \triangleright P \mapsto ([(x, y)] \cdot p \cdot M)(\nu*xvec)\langle N \rangle$ 
 $\prec P'$ 
          by(rule outputSwapFrameSubject)
        then show ?thesis
          using Pair by simp
      qed
    qed
  }
note Goal = this
obtain  $q::name prm$  where  $(q \cdot xvec) \#* yvec$  and  $(q \cdot xvec) \#* zvec$  and  $(q \cdot$ 

```

```

xvec) #* xvec
  and (q · xvec) #* N and (q · xvec) #* P' and (q · xvec) #* M
  and Sq: (set q) ⊆ (set xvec) × (set(q · xvec))
  by(rule name-list-avoiding[where xvec=xvec and c=(P, xvec, yvec, zvec, N,
M, P')]) auto
  with ⟨Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P'⟩ have Ψ ▷ P ⟶iM(ν*(q · xvec))⟨(q ·
N)⟩ < (q · P')
  by(simp add: boundOutputChainAlpha'' residualInject)
  then have (p · Ψ) ▷ P ⟶i(p · M)(ν*(q · xvec))⟨(q · N)⟩ < (q · P')
  using ⟨(q · xvec) #* M⟩ ⟨(q · xvec) #* yvec⟩ ⟨(q · xvec) #* zvec⟩
  by(rule Goal)
  with ⟨(q · xvec) #* N⟩ ⟨(q · xvec) #* P'⟩ Sq show ?thesis
  by(simp add: boundOutputChainAlpha'' residualInject)
qed

```

lemma *brouputPermFrameSubject*:

```

fixes Ψ    :: 'b
and P      :: ('a, 'b, 'c) psi
and M      :: 'a
and xvec   :: name list
and N      :: 'a
and P'     :: ('a, 'b, 'c) psi
and p      :: name prm
and yvec   :: name list
and zvec   :: name list

```

```

assumes Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P'
and S: set p ⊆ set yvec × set zvec
and yvec #* P
and zvec #* P

```

```

shows (p · Ψ) ▷ P ⟶i(p · M)(ν*xvec)⟨N⟩ < P'

```

proof –

```

{
  fix xvec N P' Xs YS
  assume Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P' and xvec #* M and xvec #* yvec and
xvec #* zvec
  have (p · Ψ) ▷ P ⟶i(p · M)(ν*xvec)⟨N⟩ < P' using S
  proof(induct p)
    case Nil
      from ⟨Ψ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P'⟩
      show ?case by simp
    next
      case(Cons a p)
        from ⟨set(a#p) ⊆ set yvec × set zvec⟩ have set p ⊆ set yvec × set zvec by
auto
        then have Trans: (p · Ψ) ▷ P ⟶i(p · M)(ν*xvec)⟨N⟩ < P' by(rule Cons)
        show ?case
        proof(cases a)

```



```

    case (Pair x y)
    note Trans
    moreover from ⟨xvec #* yvec⟩ ⟨xvec #* zvec⟩ ⟨set p ⊆ set yvec × set zvec⟩
    ⟨xvec #* M⟩ have xvec #* (p · M)
      by(simp add: freshChainSimps)
    moreover have x ∈ set yvec and y ∈ set zvec
      using ⟨set (a # p) ⊆ set yvec × set zvec⟩ Pair by auto
    with ⟨yvec #* P⟩ ⟨zvec #* P⟩ have x # P and y # P
      by(auto simp add: fresh-star-def)
    ultimately have ((x, y) · p · Ψ) ▷ P ⟶i ((x, y) · p · M)(ν*xvec)⟨N⟩
  < P'
    by(rule broutputSwapFrameSubject)
  then show ?thesis
    using Pair by simp
  qed
}
}
note Goal = this
obtain q::name prm where (q · xvec) #* yvec and (q · xvec) #* zvec and (q ·
xvec) #* xvec
  and (q · xvec) #* N and (q · xvec) #* P' and (q · xvec) #* M
  and Sq: (set q) ⊆ (set xvec) × (set(q · xvec))
  by(rule name-list-avoiding[where xvec=xvec and c=(P, xvec, yvec, zvec, N,
M, P')]) auto
with ⟨Ψ ▷ P ⟶i M(ν*xvec)⟨N⟩ < P'⟩ have Ψ ▷ P ⟶i M(ν*(q · xvec))⟨(q ·
N)⟩ < (q · P')
  by(simp add: boundOutputChainAlpha'' residualInject)
then have (p · Ψ) ▷ P ⟶i (p · M)(ν*(q · xvec))⟨(q · N)⟩ < (q · P')
  using ⟨(q · xvec) #* M⟩ ⟨(q · xvec) #* yvec⟩ ⟨(q · xvec) #* zvec⟩
  by(rule Goal)
with ⟨(q · xvec) #* N⟩ ⟨(q · xvec) #* P'⟩ Sq show ?thesis
  by(simp add: boundOutputChainAlpha'' residualInject)
qed

```

lemma *outputSwapSubject*:

```

fixes Ψ    :: 'b
and P     :: ('a, 'b, 'c) psi
and M     :: 'a
and B     :: ('a, 'b, 'c) boundOutput
and x     :: name
and y     :: name

```

assumes $\Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle < P'$

```

and xvec #* M
and x # P
and y # P
and x # Ψ
and y # Ψ

```

shows $\Psi \triangleright P \mapsto ([x, y] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([x, y] \cdot \Psi) \triangleright P \mapsto ([x, y] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
by (*rule outputSwapFrameSubject*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **show** *?thesis*
by *simp*
qed

lemma *brouputSwapSubject*:
fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c) \text{ psi}$
and M $:: 'a$
and B $:: ('a, 'b, 'c) \text{ boundOutput}$
and x $:: \text{ name}$
and y $:: \text{ name}$

assumes $\Psi \triangleright P \mapsto_i M(\nu * xvec)\langle N \rangle \prec P'$
and $xvec \#* M$
and $x \# P$
and $y \# P$
and $x \# \Psi$
and $y \# \Psi$

shows $\Psi \triangleright P \mapsto_i ([x, y] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
proof –
from $\langle \Psi \triangleright P \mapsto_i M(\nu * xvec)\langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([x, y] \cdot \Psi) \triangleright P \mapsto_i ([x, y] \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
by (*rule brouputSwapFrameSubject*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **show** *?thesis*
by *simp*
qed

lemma *outputPermSubject*:
fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c) \text{ psi}$
and M $:: 'a$
and B $:: ('a, 'b, 'c) \text{ boundOutput}$
and p $:: \text{ name prm}$
and $yvec$ $:: \text{ name list}$
and $zvec$ $:: \text{ name list}$

assumes $\Psi \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'$
and $S: \text{ set } p \subseteq \text{ set } yvec \times \text{ set } zvec$
and $yvec \#* P$
and $zvec \#* P$
and $yvec \#* \Psi$
and $zvec \#* \Psi$

shows $\Psi \triangleright P \mapsto (p \cdot M)(\nu^*xvec)\langle N \rangle \prec P'$
proof –
from *assms* **have** $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\nu^*xvec)\langle N \rangle \prec P'$
by (*metis outputPermFrameSubject*)
with $S \langle yvec \#* \Psi \rangle \langle zvec \#* \Psi \rangle$ **show** *?thesis*
by *simp*
qed

lemma *broutputPermSubject*:

fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c)$ *psi*
and M $:: 'a$
and B $:: ('a, 'b, 'c)$ *boundOutput*
and p $:: name$ *prm*
and $yvec$ $:: name$ *list*
and $zvec$ $:: name$ *list*

assumes $\Psi \triangleright P \mapsto_i M(\nu^*xvec)\langle N \rangle \prec P'$
and S : *set* $p \subseteq set\ yvec \times set\ zvec$
and $yvec \#* P$
and $zvec \#* P$
and $yvec \#* \Psi$
and $zvec \#* \Psi$

shows $\Psi \triangleright P \mapsto_i (p \cdot M)(\nu^*xvec)\langle N \rangle \prec P'$
proof –
from *assms* **have** $(p \cdot \Psi) \triangleright P \mapsto_i (p \cdot M)(\nu^*xvec)\langle N \rangle \prec P'$
by (*metis broutputPermFrameSubject*)
with $S \langle yvec \#* \Psi \rangle \langle zvec \#* \Psi \rangle$ **show** *?thesis*
by *simp*
qed

lemma *outputSwapFrame*:

fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c)$ *psi*
and M $:: 'a$
and B $:: ('a, 'b, 'c)$ *boundOutput*
and x $:: name$
and y $:: name$

assumes $\Psi \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P'$
and $xvec \#* M$
and $x \# P$
and $y \# P$
and $x \# M$
and $y \# M$

shows $([(x, y)] \cdot \Psi) \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P'$
proof –

from $\langle \Psi \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([(x, y)] \cdot \Psi) \triangleright P \mapsto ([x, y] \cdot M)(\nu * xvec) \langle N \rangle \prec P'$
by *(rule outputSwapFrameSubject)*
with $\langle x \# M \rangle \langle y \# M \rangle$ **show** *?thesis*
by *simp*
qed

lemma *brouputSwapFrame:*

fixes Ψ **::** *'b*
and P **::** *('a, 'b, 'c) psi*
and M **::** *'a*
and B **::** *('a, 'b, 'c) boundOutput*
and x **::** *name*
and y **::** *name*

assumes $\Psi \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'$
and $xvec \#* M$
and $x \# P$
and $y \# P$
and $x \# M$
and $y \# M$

shows $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'$

proof –

from $\langle \Psi \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \# P \rangle \langle y \# P \rangle$
have $([(x, y)] \cdot \Psi) \triangleright P \mapsto_i ([x, y] \cdot M)(\nu * xvec) \langle N \rangle \prec P'$
by *(rule brouputSwapFrameSubject)*
with $\langle x \# M \rangle \langle y \# M \rangle$ **show** *?thesis*
by *simp*

qed

lemma *outputPermFrame:*

fixes Ψ **::** *'b*
and P **::** *('a, 'b, 'c) psi*
and M **::** *'a*
and B **::** *('a, 'b, 'c) boundOutput*
and p **::** *name prm*
and $yvec$ **::** *name list*
and $zvec$ **::** *name list*

assumes $\Psi \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'$

and $S: set\ p \subseteq set\ yvec \times set\ zvec$
and $yvec \#* P$
and $zvec \#* P$
and $yvec \#* M$
and $zvec \#* M$

shows $(p \cdot \Psi) \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'$

proof –

from *assms* **have** $(p \cdot \Psi) \triangleright P \mapsto (p \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
by (*metis outputPermFrameSubject*)
with $S \langle yvec \#* M \rangle \langle zvec \#* M \rangle$ **show** *?thesis*
by *simp*
qed

lemma *brouputPermFrame*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $B :: ('a, 'b, 'c) \text{ boundOutput}$
and $p :: \text{ name prm}$
and $yvec :: \text{ name list}$
and $zvec :: \text{ name list}$

assumes $\Psi \triangleright P \mapsto_i M(\nu * xvec)\langle N \rangle \prec P'$
and $S: \text{ set } p \subseteq \text{ set } yvec \times \text{ set } zvec$
and $yvec \#* P$
and $zvec \#* P$
and $yvec \#* M$
and $zvec \#* M$

shows $(p \cdot \Psi) \triangleright P \mapsto_i M(\nu * xvec)\langle N \rangle \prec P'$
proof –
from *assms* **have** $(p \cdot \Psi) \triangleright P \mapsto_i (p \cdot M)(\nu * xvec)\langle N \rangle \prec P'$
by (*metis brouputPermFrameSubject*)
with $S \langle yvec \#* M \rangle \langle zvec \#* M \rangle$ **show** *?thesis*
by *simp*
qed

lemma *Comm1*:
fixes $\Psi :: 'b$
and $\Psi_Q :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c) \text{ psi}$
and $A_P :: \text{ name list}$
and $\Psi_P :: 'b$
and $Q :: ('a, 'b, 'c) \text{ psi}$
and $K :: 'a$
and $xvec :: \text{ name list}$
and $Q' :: ('a, 'b, 'c) \text{ psi}$
and $A_Q :: \text{ name list}$

assumes $\Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'$
and $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$
and $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec)\langle N \rangle \prec Q'$
and $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$

```

and  $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ 
and  $A_P \#^* \Psi$ 
and  $A_P \#^* P$ 
and  $A_P \#^* Q$ 
and  $A_P \#^* M$ 
and  $A_P \#^* A_Q$ 
and  $A_Q \#^* \Psi$ 
and  $A_Q \#^* P$ 
and  $A_Q \#^* Q$ 
and  $A_Q \#^* K$ 
and  $xvec \#^* P$ 

```

shows $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*xvec)(P' \parallel Q')$

proof –

```

{
  fix  $\Psi$  :: 'b
  and  $\Psi_Q$  :: 'b
  and  $P$  :: ('a, 'b, 'c) psi
  and  $M$  :: 'a
  and  $N$  :: 'a
  and  $P'$  :: ('a, 'b, 'c) psi
  and  $A_P$  :: name list
  and  $\Psi_P$  :: 'b
  and  $Q$  :: ('a, 'b, 'c) psi
  and  $K$  :: 'a
  and  $xvec$  :: name list
  and  $Q'$  :: ('a, 'b, 'c) psi
  and  $A_Q$  :: name list

```

```

assume  $\Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'$ 
and  $extractFrame P = \langle A_P, \Psi_P \rangle$ 
and  $distinct A_P$ 
and  $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu^*xvec)(N) \prec Q'$ 
and  $extractFrame Q = \langle A_Q, \Psi_Q \rangle$ 
and  $distinct A_Q$ 
and  $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ 
and  $A_P \#^* \Psi$ 
and  $A_P \#^* P$ 
and  $A_P \#^* Q$ 
and  $A_P \#^* M$ 
and  $A_P \#^* A_Q$ 
and  $A_Q \#^* \Psi$ 
and  $A_Q \#^* P$ 
and  $A_Q \#^* Q$ 
and  $A_Q \#^* K$ 
and  $xvec \#^* P$ 

```

have $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*xvec)(P' \parallel Q')$

proof –

obtain $r::\text{name prm}$ **where** $(r \cdot \text{xvec}) \#* \Psi$ **and** $(r \cdot \text{xvec}) \#* P$ **and** $(r \cdot \text{xvec}) \#* Q$ **and** $(r \cdot \text{xvec}) \#* M$
and $(r \cdot \text{xvec}) \#* K$ **and** $(r \cdot \text{xvec}) \#* N$ **and** $(r \cdot \text{xvec}) \#* A_P$ **and** $(r \cdot \text{xvec}) \#* A_Q$
and $(r \cdot \text{xvec}) \#* P'$ **and** $(r \cdot \text{xvec}) \#* Q'$ **and** $(r \cdot \text{xvec}) \#* \Psi_P$ **and** $(r \cdot \text{xvec}) \#* \Psi_Q$
and $Sr: (\text{set } r) \subseteq (\text{set } \text{xvec}) \times (\text{set}(r \cdot \text{xvec}))$ **and** $\text{distinctPerm } r$
by(*rule name-list-avoiding*[**where** $\text{xvec}=\text{xvec}$ **and** $c=(\Psi, P, Q, M, K, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q')$])
(auto simp add: eqvts fresh-star-prod)
obtain $q::\text{name prm}$ **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$ **and** $(q \cdot A_Q) \#* Q$ **and** $(q \cdot A_Q) \#* K$
and $(q \cdot A_Q) \#* (r \cdot N)$ **and** $(q \cdot A_Q) \#* (r \cdot \text{xvec})$ **and** $(q \cdot A_Q) \#* (r \cdot Q')$
and $(q \cdot A_Q) \#* (r \cdot P')$ **and** $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$
and $Sq: \text{set } q \subseteq \text{set } A_Q \times \text{set}(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $\text{xvec}=A_Q$ **and** $c=(\Psi, P, Q, K, r \cdot N, r \cdot \text{xvec}, \Psi_Q, A_P, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)
obtain $p::\text{name prm}$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* (r \cdot N)$ **and** $(p \cdot A_P) \#* (r \cdot \text{xvec})$ **and** $(p \cdot A_P) \#* (r \cdot Q')$
and $(p \cdot A_P) \#* (r \cdot P')$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* (q \cdot A_Q)$ **and** $Sp: (\text{set } p) \subseteq (\text{set } A_P) \times (\text{set}(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $\text{xvec}=A_P$ **and** $c=(\Psi, P, Q, M, r \cdot N, r \cdot \text{xvec}, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)
have $FrP: \text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **by fact**
have $FrQ: \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $\langle A_P \#* Q \rangle FrQ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle FrP \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)
from $\langle (r \cdot \text{xvec}) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot \text{xvec}) \rangle \langle (r \cdot \text{xvec}) \#* A_P \rangle Sp$ **have**
 $(r \cdot \text{xvec}) \#* (p \cdot A_P)$
by(*simp add: freshChainSimps*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(|N|) \prec P' \rangle Sr \langle \text{distinctPerm } r \rangle \langle \text{xvec} \#* P \rangle \langle (r \cdot \text{xvec}) \#* P \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(|(r \cdot N)|) \prec (r \cdot P')$
by(*rule inputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto (q \cdot M)(|(r \cdot N)|) \prec (r \cdot P')$ **using** Sq
 $\langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle$
by $-$ (*rule inputPermFrameSubject, (assumption | simp)+*)
then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto (q \cdot M)(|(r \cdot N)|) \prec (r \cdot P')$ **using**
 $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$

by(*simp add: eqvts*)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \text{ Sp } \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $\text{FrP}: \text{extractFrame } P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle \text{distinct } A_P \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * \text{xvec}) \langle N \rangle \prec Q' \rangle \text{ Sr } \langle (r \cdot \text{xvec}) \#* N \rangle \langle (r \cdot \text{xvec}) \#* Q' \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * (r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$
by(*simp add: boundOutputChainAlpha'' create-residual.simps*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto (p \cdot K)(\nu * (r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** $\text{Sp } \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (r \cdot \text{xvec}) \#* K \rangle \langle (r \cdot \text{xvec}) \#* A_P \rangle \langle (r \cdot \text{xvec}) \#* (p \cdot A_P) \rangle$
by(*fastforce intro: outputPermFrameSubject*)
then have $Q\text{Trans}: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (p \cdot K)(\nu * (r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** $\text{Sp } \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by(*simp add: eqvts*)
moreover then have $\text{distinct}(r \cdot \text{xvec})$ **by**(*force dest: boundOutputDistinct*)
moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \text{ Sq } \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $\text{FrQ}: \text{extractFrame } Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(q \cdot A_Q)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with $\langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle \langle (q \cdot A_Q) \#* A_P \rangle$
 $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle A_Q \#* K \rangle \langle (q \cdot A_Q) \#* K \rangle \langle A_P \#* A_Q \rangle \langle (p \cdot A_P) \#* A_Q \rangle \text{ Sp Sq}$
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$ **by**(*simp add: eqvts freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$
 Sq **have** $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* P \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* M \rangle \text{ Sq}$
have $(p \cdot A_P) \#* (q \cdot M)$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* (r \cdot N) \rangle \langle (p \cdot A_P) \#* (r \cdot P') \rangle \langle (p \cdot A_P) \#* Q \rangle$
 $\langle (p \cdot A_P) \#* (r \cdot Q') \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle (p \cdot A_P) \#* (r \cdot \text{xvec}) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \text{ Sp Sq}$ **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* (r \cdot N) \rangle \langle (q \cdot A_Q) \#* (r \cdot P') \rangle$


```

⟨(q · AQ) #* Q⟩
  moreover from ⟨(q · AQ) #* AP⟩ ⟨(p · AP) #* AQ⟩ ⟨(q · AQ) #* K⟩ ⟨(p ·
AP) #* (q · AQ)⟩ Sp Sq have (q · AQ) #* (p · K)
  by(simp add: freshChainSimps)
  moreover note ⟨(q · AQ) #* (r · Q')⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(r · xvec) #*
Ψ⟩
  moreover from ⟨(r · xvec) #* AP⟩ ⟨(p · AP) #* (r · xvec)⟩ ⟨(r · xvec) #* ΨP⟩
Sp have (r · xvec) #* (p · ΨP)
  by(simp add: freshChainSimps)
  moreover from ⟨(r · xvec) #* AQ⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(r · xvec) #* ΨQ⟩
Sq have (r · xvec) #* (q · ΨQ)
  by(simp add: freshChainSimps)
  moreover note ⟨(r · xvec) #* P⟩
  moreover from ⟨(r · xvec) #* AQ⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(r · xvec) #* M⟩
Sq have (r · xvec) #* (q · M)
  by(simp add: freshChainSimps)
  moreover note ⟨(r · xvec) #* Q⟩
  moreover from ⟨(r · xvec) #* AP⟩ ⟨(p · AP) #* (r · xvec)⟩ ⟨(r · xvec) #* K⟩
Sp have (r · xvec) #* (p · K)
  by(simp add: freshChainSimps)
  ultimately have Ψ ▷ P || Q ⟶τ < (ν*(r · xvec))((r · P') || (r · Q'))
  by - (rule cComm1)
  with ⟨(r · xvec) #* P'⟩ ⟨(r · xvec) #* Q'⟩ Sr
  show ?thesis
  by(subst resChainAlpha) auto
qed
}
note Goal = this
note ⟨Ψ ⊗ ΨQ ▷ P ⟶M(|N|) < P'⟩ ⟨Ψ ⊗ ΨP ▷ Q ⟶K(ν*xvec)|⟨N⟩ < Q'⟩
⟨Ψ ⊗ ΨP ⊗ ΨQ ⊢ M ↔ K⟩
  moreover from ⟨extractFrame P = ⟨AP, ΨP⟩⟩ ⟨AP #* Ψ⟩ ⟨AP #* P⟩ ⟨AP #* Q⟩
⟨AP #* M⟩ ⟨AP #* AQ⟩
  obtain AP' where extractFrame P = ⟨AP', ΨP⟩ and distinct AP' and AP' #*
Ψ and AP' #* P and AP' #* Q and AP' #* M and AP' #* AQ
  by - (rule distinctFrame[where C=(Ψ, P, Q, M, AQ)], auto)
  moreover from ⟨extractFrame Q = ⟨AQ, ΨQ⟩⟩ ⟨AQ #* Ψ⟩ ⟨AQ #* P⟩ ⟨AQ #*
Q⟩ ⟨AQ #* K⟩ ⟨AP' #* AQ⟩
  obtain AQ' where extractFrame Q = ⟨AQ', ΨQ⟩ and distinct AQ' and AQ' #*
Ψ and AQ' #* P and AQ' #* Q and AQ' #* K and AP' #* AQ'
  by - (rule distinctFrame[where C=(Ψ, P, Q, K, AP')], auto)
  ultimately show ?thesis using ⟨xvec #* P⟩
  by(metis Goal)
qed

lemma Comm2:
  fixes Ψ    :: 'b
  and ΨQ   :: 'b
  and P     :: ('a, 'b, 'c) psi
  and M     :: 'a

```

```

and xvec :: name list
and N    :: 'a
and P'   :: ('a, 'b, 'c) psi
and AP  :: name list
and ΨP  :: 'b
and Q    :: ('a, 'b, 'c) psi
and K    :: 'a
and Q'   :: ('a, 'b, 'c) psi
and AQ  :: name list

assumes Ψ ⊗ ΨQ ▷ P ⟶ M(ν*xvec)⟨N⟩ < P'
and   extractFrame P = ⟨AP, ΨP⟩
and   Ψ ⊗ ΨP ▷ Q ⟶ K⟨N⟩ < Q'
and   extractFrame Q = ⟨AQ, ΨQ⟩
and   Ψ ⊗ ΨP ⊗ ΨQ ⊢ M ↔ K
and   AP #* Ψ
and   AP #* P
and   AP #* Q
and   AP #* M
and   AP #* AQ
and   AQ #* Ψ
and   AQ #* P
and   AQ #* Q
and   AQ #* K
and   xvec #* Q

shows Ψ ▷ P || Q ⟶ τ < (ν*xvec)(P' || Q')
proof -
{
  fix Ψ    :: 'b
  and ΨQ  :: 'b
  and P    :: ('a, 'b, 'c) psi
  and M    :: 'a
  and xvec :: name list
  and N    :: 'a
  and P'   :: ('a, 'b, 'c) psi
  and AP  :: name list
  and ΨP  :: 'b
  and Q    :: ('a, 'b, 'c) psi
  and K    :: 'a
  and Q'   :: ('a, 'b, 'c) psi
  and AQ  :: name list

  assume Ψ ⊗ ΨQ ▷ P ⟶ M(ν*xvec)⟨N⟩ < P'
  and   extractFrame P = ⟨AP, ΨP⟩
  and   distinct AP
  and   Ψ ⊗ ΨP ▷ Q ⟶ K⟨N⟩ < Q'
  and   extractFrame Q = ⟨AQ, ΨQ⟩
  and   distinct AQ

```

and $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
and $A_P \#^* \Psi$
and $A_P \#^* P$
and $A_P \#^* Q$
and $A_P \#^* M$
and $A_P \#^* A_Q$
and $A_Q \#^* \Psi$
and $A_Q \#^* P$
and $A_Q \#^* Q$
and $A_Q \#^* K$
and $xvec \#^* Q$

have $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^* xvec)(P' \parallel Q')$
proof –

obtain $r::name\ prm$ **where** $(r \cdot xvec) \#^* \Psi$ **and** $(r \cdot xvec) \#^* P$ **and** $(r \cdot xvec) \#^* Q$ **and** $(r \cdot xvec) \#^* M$
and $(r \cdot xvec) \#^* K$ **and** $(r \cdot xvec) \#^* N$ **and** $(r \cdot xvec) \#^* A_P$ **and** $(r \cdot xvec) \#^* A_Q$
and $(r \cdot xvec) \#^* P'$ **and** $(r \cdot xvec) \#^* Q'$ **and** $(r \cdot xvec) \#^* \Psi_P$ **and** $(r \cdot xvec) \#^* \Psi_Q$
and $Sr: (set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** $distinctPerm\ r$
by(*rule name-list-avoiding*[**where** $xvec=xvec$ **and** $c=(\Psi, P, Q, M, K, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q')$])
(auto simp add: eqvts fresh-star-prod)
obtain $q::name\ prm$ **where** $(q \cdot A_Q) \#^* \Psi$ **and** $(q \cdot A_Q) \#^* P$ **and** $(q \cdot A_Q) \#^* Q$ **and** $(q \cdot A_Q) \#^* K$
and $(q \cdot A_Q) \#^* (r \cdot N)$ **and** $(q \cdot A_Q) \#^* (r \cdot xvec)$ **and** $(q \cdot A_Q) \#^* (r \cdot Q')$
and $(q \cdot A_Q) \#^* (r \cdot P')$ **and** $(q \cdot A_Q) \#^* \Psi_P$ **and** $(q \cdot A_Q) \#^* A_P$ **and** $(q \cdot A_Q) \#^* \Psi_Q$
and $Sq: set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, K, r \cdot N, r \cdot xvec, \Psi_Q, A_P, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)
obtain $p::name\ prm$ **where** $(p \cdot A_P) \#^* \Psi$ **and** $(p \cdot A_P) \#^* P$ **and** $(p \cdot A_P) \#^* Q$ **and** $(p \cdot A_P) \#^* M$
and $(p \cdot A_P) \#^* (r \cdot N)$ **and** $(p \cdot A_P) \#^* (r \cdot xvec)$ **and** $(p \cdot A_P) \#^* (r \cdot Q')$
and $(p \cdot A_P) \#^* (r \cdot P')$ **and** $(p \cdot A_P) \#^* \Psi_P$ **and** $(p \cdot A_P) \#^* \Psi_Q$ **and** $(p \cdot A_P) \#^* A_Q$
and $(p \cdot A_P) \#^* (q \cdot A_Q)$ **and** $Sp: (set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, r \cdot N, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_P \#^* Q \rangle FrQ\ \langle A_P \#^* A_Q \rangle$ **have** $A_P \#^* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)

from $\langle A_Q \#* P \rangle FrP \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)

from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* A_Q \rangle Sq$ **have**
 $(r \cdot xvec) \#* (q \cdot A_Q)$
by(*simp add: freshChainSimps*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P' \rangle Sr \langle (r \cdot xvec) \#* N \rangle \langle (r \cdot xvec) \#* P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * (r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot P')$
by(*simp add: boundOutputChainAlpha'' create-residual.simps*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto (q \cdot M)(\nu * (r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot P')$ **using** $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle (r \cdot xvec) \#* M \rangle \langle (r \cdot xvec) \#* A_Q \rangle \langle (r \cdot xvec) \#* (q \cdot A_Q) \rangle$
by(*fastforce intro: outputPermFrameSubject*)
then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto (q \cdot M)(\nu * (r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot P')$ **using** $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
by(*simp add: eqvts*)
moreover then have *distinct*($r \cdot xvec$) **by**(*force dest: boundOutputDistinct*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle distinct A_P \rangle$ **have** *distinct*($p \cdot A_P$) **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K \langle N \rangle \prec Q' \rangle Sr \langle distinctPerm r \rangle \langle xvec \#* Q \rangle \langle (r \cdot xvec) \#* Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto K \langle (r \cdot N) \rangle \prec (r \cdot Q')$
by(*rule inputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto (p \cdot K) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle$
by $- (rule inputPermFrameSubject, (assumption \mid simp) +)$
then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (p \cdot K) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by(*simp add: eqvts*)

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: extractFrame Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle distinct A_Q \rangle$ **have** *distinct*($q \cdot A_Q$) **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by(*metis chanEqClosed*)
with $\langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle \langle (q \cdot A_Q) \#* A_P \rangle$
 $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle A_Q \#* K \rangle \langle (q \cdot A_Q) \#* K \rangle \langle A_P \#* A_Q \rangle \langle (p \cdot A_P) \#* A_Q \rangle Sp Sq$

```

    have  $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$ 
      by(simp add: eqvts freshChainSimps)
    moreover note  $\langle (p \cdot A_P) \#* \Psi \rangle$ 
    moreover from  $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$ 
  Sq have  $(p \cdot A_P) \#* (q \cdot \Psi_Q)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (p \cdot A_P) \#* P \rangle$ 
    moreover from  $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* M \rangle$ 
  Sq have  $(p \cdot A_P) \#* (q \cdot M)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (p \cdot A_P) \#* (r \cdot N) \rangle \langle (p \cdot A_P) \#* (r \cdot P') \rangle \langle (p \cdot A_P) \#* Q \rangle$ 
 $\langle (p \cdot A_P) \#* (r \cdot Q') \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ 
 $\langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$ 
    moreover from  $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ 
  Sp Sq have  $(q \cdot A_Q) \#* (p \cdot \Psi_P)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* (r \cdot N) \rangle \langle (q \cdot A_Q) \#* (r \cdot P') \rangle$ 
 $\langle (q \cdot A_Q) \#* Q \rangle$ 
    moreover from  $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* K \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ 
  Sp Sq have  $(q \cdot A_Q) \#* (p \cdot K)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (q \cdot A_Q) \#* (r \cdot Q') \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi \rangle$ 
    moreover from  $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_P \rangle$ 
  Sp have  $(r \cdot xvec) \#* (p \cdot \Psi_P)$ 
    by(simp add: freshChainSimps)
    moreover from  $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_Q \rangle$ 
  Sq have  $(r \cdot xvec) \#* (q \cdot \Psi_Q)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (r \cdot xvec) \#* P \rangle$ 
    moreover from  $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* M \rangle$ 
  Sq have  $(r \cdot xvec) \#* (q \cdot M)$ 
    by(simp add: freshChainSimps)
    moreover note  $\langle (r \cdot xvec) \#* Q \rangle$ 
    moreover from  $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* K \rangle$ 
  Sp have  $(r \cdot xvec) \#* (p \cdot K)$ 
    by(simp add: freshChainSimps)
    ultimately have  $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*(r \cdot xvec))((r \cdot P') \parallel (r \cdot Q'))$ 
    by - (rule cComm2)
    with  $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$ 
  Sr
  show ?thesis
    by(subst resChainAlpha) auto
  qed
}
}
note Goal = this
note  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P' \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto K\langle N \rangle \prec Q' \rangle$ 
 $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ 
  moreover from  $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle$ 
 $\langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle$ 

```

obtain $A_{P'}$ **where** $\text{extractFrame } P = \langle A_{P'}, \Psi_P \rangle$ **and** $\text{distinct } A_{P'}$ **and** $A_{P'} \#* \Psi$ **and** $A_{P'} \#* P$ **and** $A_{P'} \#* Q$ **and** $A_{P'} \#* M$ **and** $A_{P'} \#* A_Q$
by – (rule $\text{distinctFrame}[\text{where } C=(\Psi, P, Q, M, A_Q)], \text{auto}$)
moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle A_{P'} \#* A_Q \rangle$
obtain $A_{Q'}$ **where** $\text{extractFrame } Q = \langle A_{Q'}, \Psi_Q \rangle$ **and** $\text{distinct } A_{Q'}$ **and** $A_{Q'} \#* \Psi$ **and** $A_{Q'} \#* P$ **and** $A_{Q'} \#* Q$ **and** $A_{Q'} \#* K$ **and** $A_{P'} \#* A_{Q'}$
by – (rule $\text{distinctFrame}[\text{where } C=(\Psi, P, Q, K, A_{P'})], \text{auto}$)
ultimately show $?thesis$ **using** $\langle xvec \#* Q \rangle$
by(metis Goal)
qed

lemma *BrMerge*:

fixes Ψ **::** 'b
and Ψ_Q **::** 'b
and P **::** ('a, 'b, 'c) psi
and M **::** 'a
and N **::** 'a
and P' **::** ('a, 'b, 'c) psi
and A_P **::** name list
and Ψ_P **::** 'b
and Q **::** ('a, 'b, 'c) psi
and Q' **::** ('a, 'b, 'c) psi
and A_Q **::** name list

assumes $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P'$
and $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$
and $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(|N|) \prec Q'$
and $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$
and $A_P \#* \Psi$
and $A_P \#* P$
and $A_P \#* Q$
and $A_P \#* M$
and $A_P \#* A_Q$
and $A_Q \#* \Psi$
and $A_Q \#* P$
and $A_Q \#* Q$
and $A_Q \#* M$

shows $\Psi \triangleright P \parallel Q \mapsto_i M(|N|) \prec (P' \parallel Q')$

proof –

{
fix Ψ **::** 'b
and Ψ_Q **::** 'b
and P **::** ('a, 'b, 'c) psi
and M **::** 'a
and N **::** 'a
and P' **::** ('a, 'b, 'c) psi
and A_P **::** name list

```

and  $\Psi_P$  :: 'b
and  $Q$  :: ('a, 'b, 'c) psi
and  $Q'$  :: ('a, 'b, 'c) psi
and  $A_Q$  :: name list
and  $svec$  :: name list

assume  $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P'$ 
and  $extractFrame\ P = \langle A_P, \Psi_P \rangle$ 
and  $distinct\ A_P$ 
and  $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(|N|) \prec Q'$ 
and  $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ 
and  $distinct\ A_Q$ 
and  $A_P \#^* \Psi$ 
and  $A_P \#^* P$ 
and  $A_P \#^* Q$ 
and  $A_P \#^* M$ 
and  $A_P \#^* A_Q$ 
and  $A_Q \#^* \Psi$ 
and  $A_Q \#^* P$ 
and  $A_Q \#^* Q$ 
and  $A_Q \#^* M$ 

have  $\Psi \triangleright P \parallel Q \mapsto_i M(|N|) \prec (P' \parallel Q')$ 
proof -
  obtain  $q::name\ prm$  where  $(q \cdot (A_Q::name\ list)) \#^* \Psi$  and  $(q \cdot A_Q) \#^* P$ 
    and  $(q \cdot A_Q) \#^* Q$  and  $(q \cdot A_Q) \#^* M$ 
    and  $(q \cdot A_Q) \#^* \Psi_P$  and  $(q \cdot A_Q) \#^* A_P$  and  $(q \cdot A_Q) \#^* \Psi_Q$ 
    and  $(q \cdot A_Q) \#^* N$  and  $(q \cdot A_Q) \#^* P'$  and  $(q \cdot A_Q) \#^* Q'$ 
    and  $Sq: set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$ 
    and  $distinctPerm\ q$ 
    by(rule name-list-avoiding[where  $c=(\Psi, P, M, N, P', Q', Q, \Psi_Q, A_P,$ 
 $\Psi_P)$ ]])
    (auto simp add: eqvts fresh-star-prod)
  obtain  $p::name\ prm$  where  $(p \cdot A_P) \#^* \Psi$  and  $(p \cdot A_P) \#^* P$ 
    and  $(p \cdot A_P) \#^* Q$  and  $(p \cdot A_P) \#^* M$ 
    and  $(p \cdot A_P) \#^* \Psi_P$  and  $(p \cdot A_P) \#^* \Psi_Q$  and  $(p \cdot A_P) \#^* A_Q$ 
    and  $(p \cdot A_P) \#^* N$  and  $(p \cdot A_P) \#^* P'$  and  $(p \cdot A_P) \#^* Q'$ 
    and  $S_p: set\ p \subseteq set\ A_P \times set(p \cdot A_P)$ 
    and  $(p \cdot A_P) \#^* (q \cdot A_Q)$ 
    by(rule name-list-avoiding[where  $c=(\Psi, P, N, P', Q', Q, M, A_Q, q \cdot A_Q,$ 
 $\Psi_Q, \Psi_P)$ ]])
    (auto simp add: eqvts fresh-star-prod)

have  $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$  by fact
have  $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$  by fact

from  $\langle A_P \#^* Q \rangle FrQ\ \langle A_P \#^* A_Q \rangle$  have  $A_P \#^* \Psi_Q$ 
  by(force dest: extractFrameFreshChain)
from  $\langle A_Q \#^* P \rangle FrP\ \langle A_P \#^* A_Q \rangle$  have  $A_Q \#^* \Psi_P$ 

```

by(*force dest: extractFrameFreshChain*)

from $Sq \langle A_Q \#* M \rangle \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by *simp*
from $Sp \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by *simp*

from $\langle \text{distinct } A_P \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **by** *simp*
moreover from $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(q \cdot A_Q)$ **by** *simp*
moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ $Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: \text{extractFrame } P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ $Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: \text{extractFrame } Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by(*simp add: frameChainAlpha*)

moreover have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(\downarrow N) \prec P'$ **using** $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\downarrow N) \prec P' \rangle$
by $-$ (*rule brinputPermFrameSubject, (assumption | simp)+*)
then have $(\Psi \otimes (q \cdot \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(\downarrow N) \prec P'$ **using** $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
by(*simp add: eqvts*)
with $\langle (q \cdot M) = M \rangle$ **have** $PTrans: (\Psi \otimes (q \cdot \Psi_Q)) \triangleright P \mapsto_i M(\downarrow N) \prec P'$
by *simp*

moreover have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(\downarrow N) \prec Q'$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\downarrow N) \prec Q' \rangle$
by $-$ (*rule brinputPermFrameSubject, (assumption | simp)+*)
then have $(\Psi \otimes (p \cdot \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(\downarrow N) \prec Q'$ **using** $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by(*simp add: eqvts*)
with $\langle (p \cdot M) = M \rangle$ **have** $PTrans: (\Psi \otimes (p \cdot \Psi_P)) \triangleright Q \mapsto_i M(\downarrow N) \prec Q'$
by *simp*

moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$
 Sq **have** $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by(*simp add: freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 Sp Sq **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
by(*simp add: freshChainSimps*)

moreover note

$\langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot A_P) \#* M \rangle$
 $\langle (p \cdot A_P) \#* P \rangle \langle (p \cdot A_P) \#* N \rangle \langle (p \cdot A_P) \#* P' \rangle$
 $\langle (p \cdot A_P) \#* Q \rangle \langle (p \cdot A_P) \#* Q' \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle (q \cdot A_Q) \#* \Psi \rangle \langle (q \cdot A_Q) \#* M \rangle$
 $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* N \rangle \langle (q \cdot A_Q) \#* P' \rangle$
 $\langle (q \cdot A_Q) \#* Q \rangle \langle (q \cdot A_Q) \#* Q' \rangle$


```

    ultimately show ?thesis
      by(simp add: cBrMerge)
    qed
  }
  note Goal = this

  note  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\downarrow N) \prec P' \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\downarrow N) \prec Q' \rangle$ 
  moreover from  $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#^* \Psi \rangle \langle A_P \#^* P \rangle \langle A_P \#^* Q \rangle$ 
 $\langle A_P \#^* M \rangle \langle A_P \#^* A_Q \rangle$ 
  obtain  $A_{P'}$  where  $\text{extractFrame } P = \langle A_{P'}, \Psi_P \rangle$  and distinct  $A_{P'}$  and  $A_{P'} \#^* \Psi$ 
  and  $A_{P'} \#^* P$  and  $A_{P'} \#^* Q$  and  $A_{P'} \#^* M$  and  $A_{P'} \#^* A_Q$ 
  by - (rule distinctFrame[where  $C=(\Psi, P, Q, M, A_Q)$ ], auto)
  moreover from  $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#^* \Psi \rangle \langle A_Q \#^* P \rangle \langle A_Q \#^* Q \rangle$ 
 $\langle A_Q \#^* M \rangle \langle A_{P'} \#^* A_Q \rangle$ 
  obtain  $A_{Q'}$  where  $\text{extractFrame } Q = \langle A_{Q'}, \Psi_Q \rangle$  and distinct  $A_{Q'}$  and  $A_{Q'} \#^* \Psi$ 
  and  $A_{Q'} \#^* P$  and  $A_{Q'} \#^* Q$  and  $A_{Q'} \#^* M$  and  $A_{Q'} \#^* A_Q$ 
  by - (rule distinctFrame[where  $C=(\Psi, P, Q, M, A_{P'})$ ], auto)
  ultimately show ?thesis
    by(metis Goal)
  qed

```

lemma *BrComm1*:

```

fixes  $\Psi$     :: 'b
    and  $\Psi_Q$  :: 'b
    and  $P$     :: ('a, 'b, 'c) psi
    and  $M$     :: 'a
    and  $N$     :: 'a
    and  $P'$    :: ('a, 'b, 'c) psi
    and  $A_P$   :: name list
    and  $\Psi_P$  :: 'b
    and  $Q$     :: ('a, 'b, 'c) psi
    and  $xvec$  :: name list
    and  $Q'$    :: ('a, 'b, 'c) psi
    and  $A_Q$   :: name list

```

```

assumes  $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\downarrow N) \prec P'$ 
    and  $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ 
    and  $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\downarrow \nu * xvec) \langle N \rangle \prec Q'$ 
    and  $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ 
    and  $A_P \#^* \Psi$ 
    and  $A_P \#^* P$ 
    and  $A_P \#^* Q$ 
    and  $A_P \#^* M$ 
    and  $A_P \#^* A_Q$ 
    and  $A_Q \#^* \Psi$ 
    and  $A_Q \#^* P$ 
    and  $A_Q \#^* Q$ 
    and  $A_Q \#^* M$ 
    and  $xvec \#^* P$ 

```

shows $\Psi \triangleright P \parallel Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec (P' \parallel Q')$

proof –

```
{
  fix  $\Psi$    :: 'b
  and  $\Psi_Q$  :: 'b
  and  $P$     :: ('a, 'b, 'c) psi
  and  $M$     :: 'a
  and  $N$     :: 'a
  and  $P'$    :: ('a, 'b, 'c) psi
  and  $A_P$   :: name list
  and  $\Psi_P$  :: 'b
  and  $Q$     :: ('a, 'b, 'c) psi
  and  $xvec$  :: name list
  and  $Q'$    :: ('a, 'b, 'c) psi
  and  $A_Q$   :: name list
```

```
assume  $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M \langle N \rangle \prec P'$ 
and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
and distinct  $A_P$ 
and  $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'$ 
and extractFrame  $Q = \langle A_Q, \Psi_Q \rangle$ 
and distinct  $A_Q$ 
and  $A_P \#* \Psi$ 
and  $A_P \#* P$ 
and  $A_P \#* Q$ 
and  $A_P \#* M$ 
and  $A_P \#* A_Q$ 
and  $A_Q \#* \Psi$ 
and  $A_Q \#* P$ 
and  $A_Q \#* Q$ 
and  $A_Q \#* M$ 
and  $xvec \#* P$ 
```

have $\Psi \triangleright P \parallel Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec (P' \parallel Q')$

proof –

```
obtain  $r :: name prm$  where  $(r \cdot xvec) \#* \Psi$  and  $(r \cdot xvec) \#* P$  and  $(r \cdot xvec) \#* Q$  and  $(r \cdot xvec) \#* M$ 
and  $(r \cdot xvec) \#* N$  and  $(r \cdot xvec) \#* A_P$  and  $(r \cdot xvec) \#* A_Q$ 
and  $(r \cdot xvec) \#* P'$  and  $(r \cdot xvec) \#* Q'$  and  $(r \cdot xvec) \#* \Psi_P$  and  $(r \cdot xvec) \#* \Psi_Q$ 
and  $Sr :: (set r) \subseteq (set xvec) \times (set(r \cdot xvec))$  and distinctPerm  $r$ 
by(rule name-list-avoiding[where  $xvec = xvec$  and  $c = (\Psi, P, Q, M, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q')$ ])
(auto simp add: eqvts fresh-star-prod)
obtain  $q :: name prm$  where  $(q \cdot A_Q) \#* \Psi$  and  $(q \cdot A_Q) \#* P$  and  $(q \cdot A_Q) \#* Q$  and  $(q \cdot A_Q) \#* M$ 
and  $(q \cdot A_Q) \#* (r \cdot N)$  and  $(q \cdot A_Q) \#* (r \cdot xvec)$  and  $(q \cdot A_Q) \#* (r \cdot Q')$ 
and  $(q \cdot A_Q) \#* (r \cdot P')$  and  $(q \cdot A_Q) \#* \Psi_P$  and  $(q \cdot A_Q) \#* A_P$  and  $(q$ 
```

$\cdot A_Q) \#* \Psi_Q$
and $Sq: set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, M, r \cdot N, r$
 $\cdot xvec, \Psi_Q, A_P, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)
obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P)$
 $\#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* (r \cdot N)$ **and** $(p \cdot A_P) \#* (r \cdot xvec)$ **and** $(p \cdot A_P) \#* (r \cdot Q')$
and $(p \cdot A_P) \#* (r \cdot P')$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p$
 $\cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* (q \cdot A_Q)$ **and** $Sp: (set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, r \cdot N, r$
 $\cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by fact**
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $Sp\ \langle A_P \#* M \rangle\ \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by simp
from $Sq\ \langle A_Q \#* M \rangle\ \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by simp

from $\langle A_P \#* Q \rangle\ FrQ\ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle\ FrP\ \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)
from $\langle (r \cdot xvec) \#* A_P \rangle\ \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle\ \langle (r \cdot xvec) \#* A_P \rangle\ Sp$ **have**
 $(r \cdot xvec) \#* (p \cdot A_P)$
by(*simp add: freshChainSimps*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P' \rangle\ Sr\ \langle distinctPerm\ r \rangle\ \langle xvec \#* P \rangle\ \langle (r \cdot$
 $xvec) \#* P \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|(r \cdot N)|) \prec (r \cdot P')$
by(*rule brinputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(|(r \cdot N)|) \prec (r \cdot P')$ **using** Sq
 $\langle A_Q \#* P \rangle\ \langle (q \cdot A_Q) \#* P \rangle$
by $-(rule\ brinputPermFrameSubject, (assumption\ |\ simp)+)$
then have $\Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i (q \cdot M)(|(r \cdot N)|) \prec (r \cdot P')$ **using** Sq
 $\langle A_Q \#* \Psi \rangle\ \langle (q \cdot A_Q) \#* \Psi \rangle$
by(*simp add: eqvts*)
with $\langle (q \cdot M) = M \rangle$ **have** $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i M(|(r \cdot N)|) \prec (r$
 $\cdot P')$ **by simp**

moreover from $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle\ Sp\ \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame\ P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)

moreover from $\langle \text{distinct } A_P \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu^* \text{xvec}) \langle N \rangle \prec Q' \rangle$ Sr $\langle (r \cdot \text{xvec}) \#* N \rangle \langle (r \cdot \text{xvec}) \#* Q' \rangle$

have $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$

by (*simp add: boundOutputChainAlpha'' create-residual.simps*)

then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (r \cdot \text{xvec}) \#* M \rangle \langle (r \cdot \text{xvec}) \#* A_P \rangle \langle (r \cdot \text{xvec}) \#* (p \cdot A_P) \rangle$

by (*fastforce intro: broutputPermFrameSubject*)

then have $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i (p \cdot M)(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$

using $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$

by (*simp add: eqvts*)

with $\langle (p \cdot M) = M \rangle$ **have** $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **by** *simp*

moreover then have $\text{distinct}(r \cdot \text{xvec})$ **by** (*force dest: boundOutputDistinct*)

moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ $Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$

have $FrQ: \text{extractFrame } Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$

by (*simp add: frameChainAlpha*)

moreover from $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(q \cdot A_Q)$ **by** *simp*

moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$

moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$

Sq **have** $(p \cdot A_P) \#* (q \cdot \Psi_Q)$

by (*simp add: freshChainSimps*)

moreover note $\langle (p \cdot A_P) \#* P \rangle \langle (p \cdot A_P) \#* M \rangle$

moreover note $\langle (p \cdot A_P) \#* (r \cdot N) \rangle \langle (p \cdot A_P) \#* (r \cdot P') \rangle \langle (p \cdot A_P) \#* Q \rangle$

$\langle (p \cdot A_P) \#* (r \cdot Q') \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$

$\langle (p \cdot A_P) \#* (r \cdot \text{xvec}) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$

moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ $Sp Sq$ **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$

by (*simp add: freshChainSimps*)

moreover note $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* (r \cdot N) \rangle \langle (q \cdot A_Q) \#* (r \cdot P') \rangle$

$\langle (q \cdot A_Q) \#* Q \rangle \langle (q \cdot A_Q) \#* M \rangle$

moreover note $\langle (q \cdot A_Q) \#* (r \cdot Q') \rangle \langle (q \cdot A_Q) \#* (r \cdot \text{xvec}) \rangle \langle (r \cdot \text{xvec}) \#* \Psi \rangle$

moreover from $\langle (r \cdot \text{xvec}) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot \text{xvec}) \rangle \langle (r \cdot \text{xvec}) \#* \Psi_P \rangle$

Sp **have** $(r \cdot \text{xvec}) \#* (p \cdot \Psi_P)$

by (*simp add: freshChainSimps*)

moreover from $\langle (r \cdot \text{xvec}) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot \text{xvec}) \rangle \langle (r \cdot \text{xvec}) \#* \Psi_Q \rangle$

Sq **have** $(r \cdot \text{xvec}) \#* (q \cdot \Psi_Q)$

by (*simp add: freshChainSimps*)

moreover note $\langle (r \cdot \text{xvec}) \#* P \rangle \langle (r \cdot \text{xvec}) \#* M \rangle$

moreover note $\langle (r \cdot \text{xvec}) \#* Q \rangle \langle (r \cdot \text{xvec}) \#* M \rangle$

ultimately have $\Psi \triangleright P \parallel Q \mapsto_i M(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$

by $-$ (*rule cBrComm1*)

then have *permuted*: $\Psi \triangleright P \parallel Q \mapsto_i M(\nu^*(r \cdot \text{xvec})) \langle (r \cdot N) \rangle \prec (r \cdot (P' \parallel Q'))$ **by** *simp*

```

note  $\langle (r \cdot xvec) \#* N \rangle$ 
moreover from  $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$ 
have  $\langle (r \cdot xvec) \#* (P' \parallel Q') \rangle$  by simp
moreover note Sr
moreover have set xvec  $\subseteq$  set xvec by simp
ultimately have  $\langle \nu * xvec \rangle N \prec' (P' \parallel Q') = \langle \nu * (r \cdot xvec) \rangle (r \cdot N) \prec' (r \cdot$ 
 $(P' \parallel Q'))$ 
by (rule boundOutputChainAlpha')
then have  $\langle \nu * xvec \rangle \langle N \rangle \prec (P' \parallel Q') = \langle \nu * (r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot$ 
 $(P' \parallel Q'))$ 
by (simp only: create-residual.simps)
with permuted show ?thesis
by simp
qed
}
note Goal = this

```

```

note  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \langle \nu * xvec \rangle \langle N \rangle \prec P' \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto \langle \nu * xvec \rangle \langle N \rangle \prec Q' \rangle$ 
moreover from  $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle$ 
 $\langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle$ 
obtain  $A_{P'}$  where  $extractFrame P = \langle A_{P'}, \Psi_P \rangle$  and distinct  $A_{P'}$  and  $A_{P'} \#* \Psi$ 
and  $A_{P'} \#* P$  and  $A_{P'} \#* Q$  and  $A_{P'} \#* M$  and  $A_{P'} \#* A_Q$ 
by  $- (rule distinctFrame[where C=(\Psi, P, Q, M, A_Q)], auto)$ 
moreover from  $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle$ 
 $\langle A_Q \#* M \rangle \langle A_{P'} \#* A_Q \rangle$ 
obtain  $A_{Q'}$  where  $extractFrame Q = \langle A_{Q'}, \Psi_Q \rangle$  and distinct  $A_{Q'}$  and  $A_{Q'} \#* \Psi$ 
and  $A_{Q'} \#* P$  and  $A_{Q'} \#* Q$  and  $A_{Q'} \#* M$  and  $A_{P'} \#* A_{Q'}$ 
by  $- (rule distinctFrame[where C=(\Psi, P, Q, M, A_{P'})], auto)$ 
ultimately show ?thesis using  $\langle xvec \#* P \rangle$ 
by (metis Goal)
qed

```

lemma *BrComm2:*

```

fixes  $\Psi$  :: 'b
and  $\Psi_Q$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $M$  :: 'a
and  $xvec$  :: name list
and  $N$  :: 'a
and  $P'$  :: ('a, 'b, 'c) psi
and  $A_P$  :: name list
and  $\Psi_P$  :: 'b
and  $Q$  :: ('a, 'b, 'c) psi
and  $Q'$  :: ('a, 'b, 'c) psi
and  $A_Q$  :: name list

```

```

assumes  $\Psi \otimes \Psi_Q \triangleright P \mapsto \langle \nu * xvec \rangle \langle N \rangle \prec P'$ 
and  $extractFrame P = \langle A_P, \Psi_P \rangle$ 
and  $\Psi \otimes \Psi_P \triangleright Q \mapsto \langle \nu * xvec \rangle \langle N \rangle \prec Q'$ 

```

```

and extractFrame Q = ⟨AQ, ΨQ⟩
and AP #* Ψ
and AP #* P
and AP #* Q
and AP #* M
and AP #* AQ
and AQ #* Ψ
and AQ #* P
and AQ #* Q
and AQ #* M
and xvec #* Q

```

shows $\Psi \triangleright P \parallel Q \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (P' \parallel Q')$

proof –

```

{
  fix Ψ :: 'b
  and ΨQ :: 'b
  and P :: ('a, 'b, 'c) psi
  and M :: 'a
  and xvec :: name list
  and N :: 'a
  and P' :: ('a, 'b, 'c) psi
  and AP :: name list
  and ΨP :: 'b
  and Q :: ('a, 'b, 'c) psi
  and Q' :: ('a, 'b, 'c) psi
  and AQ :: name list

```

assume $\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec P'$

```

and extractFrame P = ⟨AP, ΨP⟩
and distinct AP
and Ψ ⊗ ΨP ▷ Q ↦iM (|N|) < Q'
and extractFrame Q = ⟨AQ, ΨQ⟩
and distinct AQ
and AP #* Ψ
and AP #* P
and AP #* Q
and AP #* M
and AP #* AQ
and AQ #* Ψ
and AQ #* P
and AQ #* Q
and AQ #* M
and xvec #* Q

```

have $\Psi \triangleright P \parallel Q \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (P' \parallel Q')$

proof –

obtain $r :: \text{name prm}$ where $(r \cdot xvec) \#* \Psi$ and $(r \cdot xvec) \#* P$ and $(r \cdot xvec) \#* Q$ and $(r \cdot xvec) \#* M$

and $(r \cdot xvec) \#* M$ **and** $(r \cdot xvec) \#* N$ **and** $(r \cdot xvec) \#* A_P$ **and** $(r \cdot xvec) \#* A_Q$
and $(r \cdot xvec) \#* P'$ **and** $(r \cdot xvec) \#* Q'$ **and** $(r \cdot xvec) \#* \Psi_P$ **and** $(r \cdot xvec) \#* \Psi_Q$
and $Sr: (set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and** $distinctPerm\ r$
by(*rule name-list-avoiding*[**where** $xvec=xvec$ **and** $c=(\Psi, P, Q, M, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q')$])
(auto simp add: eqvts fresh-star-prod)
obtain $q::name\ prm$ **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$ **and** $(q \cdot A_Q) \#* Q$ **and** $(q \cdot A_Q) \#* M$
and $(q \cdot A_Q) \#* (r \cdot N)$ **and** $(q \cdot A_Q) \#* (r \cdot xvec)$ **and** $(q \cdot A_Q) \#* (r \cdot Q')$
and $(q \cdot A_Q) \#* (r \cdot P')$ **and** $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$
and $Sq: set\ q \subseteq set\ A_Q \times set(q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, M, r \cdot N, r \cdot xvec, \Psi_Q, A_P, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)
obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* (r \cdot N)$ **and** $(p \cdot A_P) \#* (r \cdot xvec)$ **and** $(p \cdot A_P) \#* (r \cdot Q')$
and $(p \cdot A_P) \#* (r \cdot P')$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* (q \cdot A_Q)$ **and** $Sp: (set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, r \cdot N, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, r \cdot Q', r \cdot P')$])
(auto simp add: eqvts fresh-star-prod)

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by fact**
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $Sp\ \langle A_P \#* M \rangle\ \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by simp
from $Sq\ \langle A_Q \#* M \rangle\ \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by simp

from $\langle A_P \#* Q \rangle\ FrQ\ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle\ FrP\ \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)
from $\langle (r \cdot xvec) \#* A_Q \rangle\ \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle\ \langle (r \cdot xvec) \#* A_Q \rangle\ Sq$ **have**
 $(r \cdot xvec) \#* (q \cdot A_Q)$
by(*simp add: freshChainSimps*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \rangle \langle N \rangle \prec P'$ $Sr\ \langle (r \cdot xvec) \#* N \rangle\ \langle (r \cdot xvec) \#* P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * (r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot P')$
by(*simp add: boundOutputChainAlpha'' create-residual.simps*)

then have $\langle q \cdot (\Psi \otimes \Psi_Q) \triangleright P \mapsto_i (q \cdot M) \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot P') \rangle$ **using** $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle (r \cdot xvec) \#* M \rangle \langle (r \cdot xvec) \#* A_Q \rangle \langle (r \cdot xvec) \#* (q \cdot A_Q) \rangle$
by *(fastforce intro: broutputPermFrameSubject)*
then have $\Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i (q \cdot M) \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot P')$
using $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
by *(simp add: eqvts)*
with $\langle (q \cdot M) = M \rangle$ **have** $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i M \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot P')$ **by** *simp*
moreover then have $distinct(r \cdot xvec)$ **by** *(force dest: boundOutputDistinct)*

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by *(simp add: frameChainAlpha)*
moreover from $\langle distinct A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M \langle N \rangle \prec Q' \rangle Sr \langle distinctPerm r \rangle \langle xvec \#* Q \rangle \langle (r \cdot xvec) \#* Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M \langle (r \cdot N) \rangle \prec (r \cdot Q')$
by *(rule brinputAlpha)*
then have $\langle (p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M) \langle (r \cdot N) \rangle \prec (r \cdot Q') \rangle$ **using** $Sq \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle$
by *– (rule brinputPermFrameSubject, (assumption | simp)+)*
then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i (p \cdot M) \langle (r \cdot N) \rangle \prec (r \cdot Q')$
using $Sq \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by *(simp add: eqvts)*
with $\langle (p \cdot M) = M \rangle$ **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M \langle (r \cdot N) \rangle \prec (r \cdot Q')$
by *simp*

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: extractFrame Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by *(simp add: frameChainAlpha)*
moreover from $\langle distinct A_Q \rangle$ **have** $distinct(q \cdot A_Q)$ **by** *simp*

moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$
 Sq **have** $\langle (p \cdot A_P) \#* (q \cdot \Psi_Q) \rangle$
by *(simp add: freshChainSimps)*
moreover note $\langle (p \cdot A_P) \#* P \rangle \langle (p \cdot A_P) \#* M \rangle$
moreover note $\langle (p \cdot A_P) \#* (r \cdot N) \rangle \langle (p \cdot A_P) \#* (r \cdot P') \rangle \langle (p \cdot A_P) \#* Q \rangle$
 $\langle (p \cdot A_P) \#* (r \cdot Q') \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle Sp Sq$ **have** $\langle (q \cdot A_Q) \#* (p \cdot \Psi_P) \rangle$
by *(simp add: freshChainSimps)*
moreover note $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* (r \cdot N) \rangle \langle (q \cdot A_Q) \#* (r \cdot P') \rangle$
 $\langle (q \cdot A_Q) \#* Q \rangle \langle (q \cdot A_Q) \#* M \rangle$
moreover note $\langle (q \cdot A_Q) \#* (r \cdot Q') \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi \rangle$

moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_P \rangle$
Sp **have** $\langle (r \cdot xvec) \#* (p \cdot \Psi_P) \rangle$
by (*simp add: freshChainSimps*)
moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_Q \rangle$
Sq **have** $\langle (r \cdot xvec) \#* (q \cdot \Psi_Q) \rangle$
by (*simp add: freshChainSimps*)
moreover note $\langle (r \cdot xvec) \#* P \rangle \langle (r \cdot xvec) \#* M \rangle$
moreover note $\langle (r \cdot xvec) \#* Q \rangle \langle (r \cdot xvec) \#* M \rangle$
ultimately have $\Psi \triangleright P \parallel Q \mapsto_i M(\nu^*(r \cdot xvec)) \langle (r \cdot N) \rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$
by – (*rule cBrComm2*)
then have permuted: $\Psi \triangleright P \parallel Q \mapsto_i M(\nu^*(r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot (P' \parallel Q'))$
by *simp*
note $\langle (r \cdot xvec) \#* N \rangle$
moreover from $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$
have $\langle (r \cdot xvec) \#* (P' \parallel Q') \rangle$ **by** *simp*
moreover note *Sr*
moreover have *set xvec* \subseteq *set xvec* **by** *simp*
ultimately have $(\nu^*xvec)N \prec' (P' \parallel Q') = (\nu^*(r \cdot xvec)) \langle (r \cdot N) \rangle \prec' (r \cdot (P' \parallel Q'))$
by (*rule boundOutputChainAlpha''*)
then have $iM(\nu^*xvec) \langle N \rangle \prec (P' \parallel Q') = iM(\nu^*(r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot (P' \parallel Q'))$
by (*simp only: create-residual.simps*)
with permuted show *?thesis*
by *simp*
qed
}
note *Goal = this*

note $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu^*xvec) \langle N \rangle \prec P' \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q' \rangle$
moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle$
obtain $A_{P'}$ **where** $extractFrame P = \langle A_{P'}, \Psi_P \rangle$ **and** *distinct* $A_{P'}$ **and** $A_{P'} \#* \Psi$ **and** $A_{P'} \#* P$ **and** $A_{P'} \#* Q$ **and** $A_{P'} \#* M$ **and** $A_{P'} \#* A_Q$
by – (*rule distinctFrame[where C=(Ψ, P, Q, M, A_Q)], auto*)
moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle A_{P'} \#* A_Q \rangle$
obtain $A_{Q'}$ **where** $extractFrame Q = \langle A_{Q'}, \Psi_Q \rangle$ **and** *distinct* $A_{Q'}$ **and** $A_{Q'} \#* \Psi$ **and** $A_{Q'} \#* P$ **and** $A_{Q'} \#* Q$ **and** $A_{Q'} \#* M$ **and** $A_{P'} \#* A_{Q'}$
by – (*rule distinctFrame[where C=(Ψ, P, Q, M, A_{P'})], auto*)
ultimately show *?thesis using* $\langle xvec \#* Q \rangle$
by (*metis Goal*)
qed

lemma *BrClose:*

fixes Ψ **::** *'b*
and P **::** (*'a, 'b, 'c*) *psi*
and M **::** *'a*

```

and  $xvec$  :: name list
and  $N$     :: 'a
and  $P'$    :: ('a, 'b, 'c) psi
and  $x$     :: name

assumes  $\Psi \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P'$ 
and  $x \in \text{supp } M$ 
and  $x \# \Psi$ 

shows  $\Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)((\nu*xvec)P')$ 
proof -
obtain  $p$  where  $xvecFreshPsi$ :  $((p::name\ perm) \cdot (xvec::name\ list)) \#* \Psi$ 
and  $xvecFreshM$ :  $(p \cdot xvec) \#* M$ 
and  $xvecFreshN$ :  $(p \cdot xvec) \#* N$ 
and  $xvecFreshP$ :  $(p \cdot xvec) \#* P$ 
and  $xvecFreshP'$ :  $(p \cdot xvec) \#* P'$ 
and  $xvecFreshx$ :  $(p \cdot xvec) \#* x$ 
and  $S$ :  $(\text{set } p) \subseteq (\text{set } xvec) \times (\text{set}(p \cdot xvec))$ 
and  $dp$ :  $\text{distinctPerm } p$ 
by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(\Psi, M, N, P, P', x)$ ])
  (auto simp add: eqvts fresh-star-prod)

obtain  $y::name$  where  $y \# P$  and  $y \# xvec$  and  $y \neq x$  and  $y \# N$ 
and  $y \# (p \cdot xvec)$  and  $y \# (p \cdot P')$ 
and  $y \# M$  and  $y \# \Psi$  and  $y \# P'$ 
by(generate-fresh name) (auto simp add: freshChainSimps)

from  $\langle y \# (p \cdot xvec) \rangle \langle y \# (p \cdot P') \rangle$ 
have  $yFreshRes$ :  $y \# ((\nu*(p \cdot xvec))(p \cdot P'))$ 
  by(simp add: resChainFresh)

from  $\langle \Psi \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P' \rangle S$ 
   $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P' \rangle$ 
have  $\Psi \triangleright P \mapsto iM(\nu*(p \cdot xvec))\langle (p \cdot N) \rangle \prec (p \cdot P')$ 
  by(simp add: alphaOutputResidual)

then have  $[(x, y)] \cdot (\Psi \triangleright P \mapsto iM(\nu*(p \cdot xvec))\langle (p \cdot N) \rangle \prec (p \cdot P'))$ 
  by simp
with  $\langle (p \cdot xvec) \#* x \rangle \langle y \# (p \cdot xvec) \rangle$ 
   $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ 
have  $pretrans$ :  $\Psi \triangleright ([x, y] \cdot P) \mapsto i([x, y] \cdot M)(\nu*(p \cdot xvec))\langle ([x, y] \cdot (p \cdot N)) \rangle \prec ([x, y] \cdot (p \cdot P'))$ 
  by(simp add: eqvts)

moreover from  $\langle x \in \text{supp } M \rangle \langle y \# M \rangle$ 
have  $y \in \text{supp } ([x, y] \cdot M)$ 
  by (metis fresh-bij fresh-def swap-simps)

moreover from  $pretrans$ 

```

have *distinct* $\langle p \cdot xvec \rangle$
by(*force dest: boundOutputDistinct*)

moreover note $\langle (p \cdot xvec) \#* \Psi \rangle$
moreover from $\langle (p \cdot xvec) \#* P \rangle \langle (p \cdot xvec) \#* x \rangle \langle y \# (p \cdot xvec) \rangle$
have $\langle (p \cdot xvec) \#* [(x, y)] \cdot P \rangle$ **by** *simp*
moreover from $\langle (p \cdot xvec) \#* M \rangle \langle (p \cdot xvec) \#* x \rangle \langle y \# (p \cdot xvec) \rangle$
have $\langle (p \cdot xvec) \#* [(x, y)] \cdot M \rangle$ **by** *simp*
moreover note $\langle y \# \Psi \rangle \langle y \# (p \cdot xvec) \rangle$

ultimately have $\Psi \triangleright \langle \nu y \rangle([(x, y)] \cdot P) \longmapsto \tau \prec \langle \nu y \rangle(\langle \nu*(p \cdot xvec) \rangle([(x, y)] \cdot (p \cdot P')))$
by(*rule cBrClose*)
with $\langle (p \cdot xvec) \#* x \rangle \langle y \# (p \cdot xvec) \rangle$
have $\Psi \triangleright \langle \nu y \rangle([(x, y)] \cdot P) \longmapsto \tau \prec \langle \nu y \rangle(\langle \nu*(p \cdot xvec) \rangle([(x, y)] \cdot (\langle \nu*(p \cdot xvec) \rangle(p \cdot P'))))$
by(*simp add: eqvts*)
with *yFreshRes* $\langle y \# P \rangle$
have $\Psi \triangleright \langle \nu x \rangle P \longmapsto \tau \prec \langle \nu x \rangle(\langle \nu*(p \cdot xvec) \rangle(p \cdot P'))$
by(*simp add: alphaRes*)

with $\langle (p \cdot xvec) \#* P' \rangle S$
show *?thesis*
by(*simp add: resChainAlpha*)

qed

lemma *semanticsCasesAux*[*consumes 1, case-names cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:

fixes $cP :: ('a, 'b, 'c) psi$
and $cRs :: ('a, 'b, 'c) residual$
and $C :: 'f::fs-name$
and $x :: name$

assumes $\Psi \triangleright cP \longmapsto cRs$
and *rInput*: $\bigwedge M K xvec N Tvec P. \llbracket cP = M(\lambda*xvec N).P; cRs = K(\langle N[xvec::=Tvec] \rangle) \prec P[xvec::=Tvec];$

$\Psi \vdash M \leftrightarrow K; distinct\ xvec; set\ xvec \subseteq supp\ N;$
 $length\ xvec=length\ Tvec;$

$xvec \#* Tvec; xvec \#* \Psi; xvec \#* M; xvec \#* K;$

$xvec \#* C \Longrightarrow Prop$

and *rBrInput*: $\bigwedge M K xvec N Tvec P. \llbracket cP = M(\lambda*xvec N).P; cRs = \iota K(\langle N[xvec::=Tvec] \rangle) \prec P[xvec::=Tvec];$

$\Psi \vdash K \succeq M; distinct\ xvec; set\ xvec \subseteq supp\ N;$
 $length\ xvec=length\ Tvec;$

$xvec \#* Tvec; xvec \#* \Psi; xvec \#* M; xvec \#* K;$

$xvec \#* C \Longrightarrow Prop$

and *rOutput*: $\bigwedge M K N P. \llbracket cP = M\langle N \rangle.P; cRs = K\langle N \rangle \prec P; \Psi \vdash M \leftrightarrow K \rrbracket$
 $\Longrightarrow Prop$

and *rBrOutput*: $\bigwedge M K N P. \llbracket cP = M\langle N \rangle.P; cRs = \iota K\langle N \rangle \prec P; \Psi \vdash M \preceq$

$K \parallel \Longrightarrow Prop$
and $rCase: \bigwedge Cs P \varphi. \llbracket cP = Cases Cs; \Psi \triangleright P \mapsto cRs; (\varphi, P) \in set Cs; \Psi \vdash \varphi; guarded P \rrbracket \Longrightarrow Prop$

and $rPar1: \bigwedge \Psi_Q P \alpha P' Q A_Q. \llbracket cP = P \parallel Q; cRs = \alpha \prec (P' \parallel Q); (\Psi \otimes \Psi_Q) \triangleright P \mapsto (\alpha \prec P'); extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_Q \#* P; A_Q \#* Q; A_Q \#* \Psi; A_Q \#* \alpha; A_Q \#* C; A_Q \#* P'; bn \alpha \#* \Psi; bn \alpha \#* \Psi_Q;$
 $bn \alpha \#* Q; bn \alpha \#* P; bn \alpha \#* subject \alpha; bn \alpha \#* C; distinct(bn \alpha) \rrbracket \Longrightarrow$
 $Prop$

and $rPar2: \bigwedge \Psi_P Q \alpha Q' P A_P. \llbracket cP = P \parallel Q; cRs = \alpha \prec (P \parallel Q'); (\Psi \otimes \Psi_P) \triangleright Q \mapsto \alpha \prec Q'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* P; A_P \#* Q; A_P \#* \Psi; A_P \#* \alpha; A_P \#* C;$
 $A_P \#* Q'; bn \alpha \#* \Psi; bn \alpha \#* \Psi_P; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* subject \alpha; bn \alpha \#* C; distinct(bn \alpha) \rrbracket \Longrightarrow Prop$

and $rComm1: \bigwedge \Psi_Q P M N P' A_P \Psi_P Q K xvec Q' A_Q. \llbracket cP = P \parallel Q; cRs = \tau \prec (\nu * xvec) P' \parallel Q'; \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec) \langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N;$
 $A_P \#* P'; A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi;$
 $A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; A_Q \#* xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#* Q;$
 $xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C; distinct xvec \rrbracket \Longrightarrow Prop$

and $rComm2: \bigwedge \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q. \llbracket cP = P \parallel Q; cRs = \tau \prec (\nu * xvec) P' \parallel Q'; \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N;$
 $A_P \#* P'; A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi;$
 $A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; A_Q \#* xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#* Q;$
 $xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C; distinct xvec \rrbracket \Longrightarrow Prop$

and $rBrMerge: \bigwedge \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q.$

$\llbracket cP = (P \parallel Q); cRs = \iota M(\langle N \rangle) \prec (P' \parallel Q');$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(\langle N \rangle) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
distinct A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\langle N \rangle) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
distinct A_Q ;
 $A_P \# \Psi; A_P \# \Psi_Q; A_P \# P; A_P \# N; A_P \# P';$
 $A_P \# Q; A_P \# Q'; A_P \# M; A_P \# A_Q;$
 $A_Q \# \Psi; A_Q \# \Psi_P; A_Q \# P; A_Q \# N; A_Q \# P';$
 $A_Q \# Q; A_Q \# Q'; A_Q \# M; A_P \# C; A_Q \# C \rrbracket \implies \text{Prop}$
and *rBrComm1*: $\bigwedge \Psi_Q P M N P' A_P \Psi_P Q \text{vec } Q' A_Q.$
 $\llbracket cP = P \parallel Q; cRs = \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \prec (P' \parallel Q');$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(\langle N \rangle) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
distinct A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \prec Q'; \text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle;$ *distinct* A_Q ;
 $A_P \# \Psi; A_P \# \Psi_Q; A_P \# P; A_P \# N;$
 $A_P \# P'; A_P \# Q; A_P \# Q'; A_P \# M; A_P \# A_Q; A_P \# \text{vec};$
 $A_Q \# \Psi; A_Q \# \Psi_P;$
 $A_Q \# P; A_Q \# N; A_Q \# P'; A_Q \# M; A_Q \# Q; A_Q \# Q'; A_Q$
 $\# \text{vec};$
 $\text{vec} \# \Psi; \text{vec} \# \Psi_P; \text{vec} \# \Psi_Q; \text{vec} \# P; \text{vec} \# Q; \text{vec} \#$
 $M;$
 $A_P \# C; A_Q \# C; \text{vec} \# C; \text{distinct } \text{vec} \rrbracket \implies \text{Prop}$
and *rBrComm2*: $\bigwedge \Psi_Q P M \text{vec } N P' A_P \Psi_P Q Q' A_Q.$
 $\llbracket cP = P \parallel Q; cRs = \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \prec (P' \parallel Q');$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \prec P'; \text{extractFrame } P = \langle A_P,$
 $\Psi_P \rangle;$ *distinct* A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\langle N \rangle) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
distinct A_Q ;
 $A_P \# \Psi; A_P \# \Psi_Q; A_P \# P; A_P \# N;$
 $A_P \# P'; A_P \# Q; A_P \# Q'; A_P \# M; A_P \# A_Q; A_P \# \text{vec};$
 $A_Q \# \Psi; A_Q \# \Psi_P;$
 $A_Q \# P; A_Q \# N; A_Q \# P'; A_Q \# M; A_Q \# Q; A_Q \# Q'; A_Q$
 $\# \text{vec};$
 $\text{vec} \# \Psi; \text{vec} \# \Psi_P; \text{vec} \# \Psi_Q; \text{vec} \# P; \text{vec} \# Q; \text{vec} \#$
 $M;$
 $A_P \# C; A_Q \# C; \text{vec} \# C; \text{distinct } \text{vec} \rrbracket \implies \text{Prop}$
and *rBrClose*: $\bigwedge P M \text{vec } N P' x.$
 $\llbracket cP = (\nu x)P; cRs = \tau \prec (\nu x)(\langle \nu * \text{vec} \rangle P');$
 $x \in \text{supp } M;$
 $\Psi \triangleright P \mapsto \iota M(\langle \nu * \text{vec} \rangle \langle N \rangle) \prec P';$
distinct $\text{vec}; \text{vec} \# \Psi; \text{vec} \# P;$
 $\text{vec} \# M;$
 $x \# \Psi; x \# \text{vec};$
 $\text{vec} \# C; x \# C \rrbracket \implies \text{Prop}$
and *rOpen*: $\bigwedge P M \text{vec } yvec N P' x.$
 $\llbracket cP = (\nu x)P; cRs = M(\langle \nu * (\text{vec} @ x \# yvec) \rangle \langle N \rangle) \prec P';$
 $\Psi \triangleright P \mapsto M(\langle \nu * (\text{vec} @ yvec) \rangle \langle N \rangle) \prec P'; x \in \text{supp } N; x \# \text{vec}; x \#$
 $yvec; x \# M; x \# \Psi; \text{distinct } \text{vec}; \text{distinct } yvec;$

$xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* yvec; yvec \#* \Psi; yvec \#* P;$
 $yvec \#* M; xvec \#* C; x \# C; yvec \#* C \implies$
Prop
and *rBrOpen*: $\bigwedge P M xvec yvec N P' x.$
 $\llbracket cP = (\nu x)P; cRs = \imath M(\nu*(xvec@x\#yvec))\langle N \rangle \prec P';$
 $\Psi \triangleright P \mapsto \imath M(\nu*(xvec@yvec))\langle N \rangle \prec P'; x \in \text{supp } N; x \# xvec; x \#$
 $yvec; x \# M; x \# \Psi; \text{distinct } xvec; \text{distinct } yvec;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* yvec; yvec \#* \Psi; yvec \#* P;$
 $yvec \#* M; xvec \#* C; x \# C; yvec \#* C \implies$
Prop
and *rScope*: $\bigwedge P \alpha P' x. \llbracket cP = (\nu x)P; cRs = \alpha \prec (\nu x)P';$
 $\Psi \triangleright P \mapsto \alpha \prec P'; x \# \Psi; x \# \alpha; x \# C; \text{bn } \alpha \#* \Psi; \text{bn } \alpha$
 $\#* P; \text{bn } \alpha \#* \text{subject } \alpha; \text{bn } \alpha \#* C; \text{distinct}(\text{bn } \alpha) \rrbracket \implies \text{Prop}$
and *rBang*: $\bigwedge P. \llbracket cP = !P; \Psi \triangleright P \parallel !P \mapsto cRs; \text{guarded } P \rrbracket \implies \text{Prop}$
shows *Prop*
using $\langle \Psi \triangleright cP \mapsto cRs \rangle$
proof(*cases rule: semantics.cases*)
case(*cInput M K xvec N Tvec P*)
obtain $p::\text{name prm}$ **where** $(p \cdot xvec) \#* \Psi$ **and** $(p \cdot xvec) \#* M$ **and** $(p \cdot xvec)$
 $\#* N$ **and** $(p \cdot xvec) \#* K$
and $(p \cdot xvec) \#* Tvec$ **and** $(p \cdot xvec) \#* P$ **and** $(p \cdot xvec) \#* C$
and $S: (\text{set } p) \subseteq (\text{set } xvec) \times (\text{set}(p \cdot xvec))$ **and** *distinctPerm p*
by(*rule name-list-avoiding[where xvec=xvec and c=(\Psi, M, K, N, P, C,*
Tvec)])
(auto simp add: eqvts fresh-star-prod)
from $\langle cP = M(\lambda*xvec N).P \rangle \langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P \rangle S$
have $cP = M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot P)$
by(*simp add: inputChainAlpha'*)
moreover from $\langle cRs = K(\langle N[xvec::=Tvec] \rangle) \prec P[xvec::=Tvec] \rangle \langle (p \cdot xvec) \#*$
 $N \rangle \langle (p \cdot xvec) \#* P \rangle S \langle \text{length } xvec = \text{length } Tvec \rangle \langle \text{distinctPerm } p \rangle$
have $cRs = K(\langle (p \cdot N)[(p \cdot xvec)::=Tvec] \rangle) \prec (p \cdot P)[(p \cdot xvec)::=Tvec]$
by(*auto simp add: substTerm.renaming renaming residualInject*)

moreover note $\langle \Psi \vdash M \leftrightarrow K \rangle$
moreover from $\langle \text{distinct } xvec \rangle$ **have** *distinct*($p \cdot xvec$)
by *simp*
moreover from $\langle (\text{set } xvec) \subseteq (\text{supp } N) \rangle$ **have** $(p \cdot (\text{set } xvec)) \subseteq (p \cdot (\text{supp } N))$
by *simp*
then have $\text{set}(p \cdot xvec) \subseteq \text{supp}(p \cdot N)$
by(*simp add: eqvts*)
moreover from $\langle \text{length } xvec = \text{length } Tvec \rangle$ **have** $\text{length}(p \cdot xvec) = \text{length } Tvec$
by *simp*
ultimately show *?thesis* **using** $\langle (p \cdot xvec) \#* Tvec \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle (p \cdot xvec)$
 $\#* M \rangle \langle (p \cdot xvec) \#* K \rangle$
 $\langle (p \cdot xvec) \#* C \rangle$
by(*rule rInput*)
next
case(*cBrInput K M xvec N Tvec P*)
obtain $p::\text{name prm}$ **where** $(p \cdot xvec) \#* \Psi$ **and** $(p \cdot xvec) \#* M$ **and** $(p \cdot xvec)$

```

#* N and (p · xvec) #* K
  and (p · xvec) #* Tvec and (p · xvec) #* P and (p · xvec) #* C
  and S: (set p) ⊆ (set xvec) × (set(p · xvec)) and distinctPerm p
  by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, M, K, N, P, C,
Tvec)])
  (auto simp add: eqvts fresh-star-prod)
  from ⟨cP = M(λ*xvec N).P⟩ ⟨(p · xvec) #* N⟩ ⟨(p · xvec) #* P⟩ S
  have cP = M(λ*(p · xvec) (p · N)).(p · P)
  by(simp add: inputChainAlpha')
  moreover from ⟨cRs = ιK((N[xvec::=Tvec]) ↦ P[xvec::=Tvec])⟩ ⟨(p · xvec) #*
N⟩ ⟨(p · xvec) #* P⟩ S ⟨length xvec = length Tvec⟩ ⟨distinctPerm p⟩
  have cRs = ιK(((p · N)[(p · xvec)::=Tvec]) ↦ (p · P)[(p · xvec)::=Tvec])
  by(auto simp add: substTerm.renaming renaming residualInject)

  moreover note ⟨Ψ ⊢ K ≥ M⟩
  moreover from ⟨distinct xvec⟩ have distinct(p · xvec)
  by simp
  moreover from ⟨(set xvec) ⊆ (supp N)⟩ have (p · (set xvec)) ⊆ (p · (supp N))
  by simp
  then have set(p · xvec) ⊆ supp(p · N)
  by(simp add: eqvts)
  moreover from ⟨length xvec = length Tvec⟩ have length(p · xvec) = length Tvec
  by simp
  ultimately show ?thesis using ⟨(p · xvec) #* Tvec⟩ ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec)
#* M⟩ ⟨(p · xvec) #* K⟩
  ⟨(p · xvec) #* C⟩
  by(simp add: rBrInput)
next
  case(Output M K N P)
  then show ?thesis by(rule rOutput)
next
  case(BrOutput M K N P)
  then show ?thesis by(rule rBrOutput)
next
  case(Case P φ Cs)
  then show ?thesis by(rule rCase)
next
  case(cPar1 ΨQ P α P' Q AQ)
  obtain q::name prm where (bn(q · α)) #* Ψ and (bn(q · α)) #* P and (bn(q ·
α)) #* Q
  and (bn(q · α)) #* α and (bn(q · α)) #* AQ and (bn(q · α)) #* P' and (bn(q
· α)) #* ΨQ
  and distinctPerm q
  and (bn(q · α)) #* C and Sq: (set q) ⊆ set(bn α) × (set(bn(q · α)))
  by(rule name-list-avoiding[where xvec=bn α and c=(Ψ, P, Q, α, AQ, ΨQ, P',
C)]) (auto simp add: eqvts)
  obtain p::name prm where (p · AQ) #* Ψ and (p · AQ) #* P and (p · AQ) #*
Q
  and (p · AQ) #* α and (p · AQ) #* (q · α) and (p · AQ) #* P'

```

and $\langle p \cdot A_Q \rangle \#* \langle q \cdot P' \rangle$ **and** $\langle p \cdot A_Q \rangle \#* \Psi_Q$ **and** $\langle p \cdot A_Q \rangle \#* C$
and $Sp: (\text{set } p) \subseteq (\text{set } A_Q) \times (\text{set}(p \cdot A_Q))$ **and** $\text{distinctPerm } p$
by($\text{rule name-list-avoiding}[\text{where } xvec=A_Q \text{ and } c=(\Psi, P, Q, \alpha, q \cdot \alpha, P', (q \cdot P'), \Psi_Q, C)]$) *auto*
from $\langle A_Q \rangle \#* \alpha$ $\langle bn(q \cdot \alpha) \rangle \#* A_Q$ $\langle Sq \rangle \langle \text{distinctPerm } q \rangle$ **have** $A_Q \#* \langle q \cdot \alpha \rangle$
by($\text{subst fresh-star-bij}[\text{symmetric, of - - } q]$) (*simp add: eqvts*)
from $\langle bn \alpha \rangle \#* \text{subject } \alpha$ $\langle \text{distinctPerm } q \rangle$ **have** $bn(q \cdot \alpha) \#* \text{subject}(q \cdot \alpha)$
by($\text{subst fresh-star-bij}[\text{symmetric, of - - } q]$) (*simp add: eqvts*)
from $\langle \text{distinct}(bn \alpha) \rangle$ $\langle \text{distinctPerm } q \rangle$ **have** $\text{distinct}(bn(q \cdot \alpha))$
by($\text{subst distinctClosed}[\text{symmetric, of - } q]$) (*simp add: eqvts*)
note $\langle cP = P \parallel Q \rangle$

moreover from $\langle cRs = \alpha \prec (P' \parallel Q) \rangle$ $\langle bn \alpha \rangle \#* \text{subject } \alpha$ $\langle (bn(q \cdot \alpha)) \rangle \#* \alpha$
 $\langle (bn(q \cdot \alpha)) \rangle \#* P'$ $\langle (bn(q \cdot \alpha)) \rangle \#* Q$ $\langle bn \alpha \rangle \#* Q$ $\langle Sq \rangle$
have $cRs = (q \cdot \alpha) \prec (q \cdot P') \parallel Q$
by(*force simp add: residualAlpha*)
moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle$ $\langle bn \alpha \rangle \#* \text{subject } \alpha$ $\langle (bn(q \cdot \alpha)) \rangle \#*$
 α $\langle (bn(q \cdot \alpha)) \rangle \#* P'$ $\langle Sq \rangle$
have $\text{Trans}: \Psi \otimes \Psi_Q \triangleright P \mapsto (q \cdot \alpha) \prec (q \cdot P')$
by(*force simp add: residualAlpha*)
then have $A_Q \#* \langle q \cdot P' \rangle$ **using** $\langle bn(q \cdot \alpha) \rangle \#* \text{subject}(q \cdot \alpha)$ $\langle \text{distinct}(bn(q \cdot \alpha)) \rangle$
 $\langle A_Q \rangle \#* P$ $\langle A_Q \rangle \#* \langle q \cdot \alpha \rangle$
by(*force dest: freeFreshChainDerivative*)

from Trans **have** $\langle p \cdot (\Psi \otimes \Psi_Q) \rangle \triangleright \langle p \cdot P \rangle \mapsto p \cdot ((q \cdot \alpha) \prec (q \cdot P'))$
by(*rule semantics.eqvt*)
with $\langle A_Q \rangle \#* \Psi$ $\langle A_Q \rangle \#* P$ $\langle A_Q \rangle \#* \langle q \cdot \alpha \rangle$ $\langle A_Q \rangle \#* \langle q \cdot P' \rangle$ $\langle (bn(q \cdot \alpha)) \rangle \#* A_Q$
 $\langle p \cdot A_Q \rangle \#* \langle q \cdot \alpha \rangle$
 $\langle p \cdot A_Q \rangle \#* \Psi$ $\langle p \cdot A_Q \rangle \#* P$ $\langle p \cdot A_Q \rangle \#* \langle q \cdot P' \rangle$ $\langle Sp \rangle$
have $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto (q \cdot \alpha) \prec (q \cdot P')$ **by**(*simp add: eqvts*)
moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ $\langle p \cdot A_Q \rangle \#* \Psi_Q$ $\langle Sp \rangle$ **have**
 $\text{extractFrame } Q = \langle (p \cdot A_Q), (p \cdot \Psi_Q) \rangle$
by(*simp add: frameChainAlpha' eqvts*)
moreover from $\langle (bn(q \cdot \alpha)) \rangle \#* \Psi_Q$ $\langle (bn(q \cdot \alpha)) \rangle \#* A_Q$ $\langle p \cdot A_Q \rangle \#* \langle q \cdot \alpha \rangle$
 $\langle Sp \rangle$ **have** $\langle bn(q \cdot \alpha) \rangle \#* \langle p \cdot \Psi_Q \rangle$
by(*simp add: freshAlphaPerm*)
moreover from $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_Q)$ **by** *simp*
ultimately show *?thesis*
using $\langle p \cdot A_Q \rangle \#* P$ $\langle p \cdot A_Q \rangle \#* Q$ $\langle p \cdot A_Q \rangle \#* \Psi$ $\langle p \cdot A_Q \rangle \#* \langle q \cdot \alpha \rangle$
 $\langle p \cdot A_Q \rangle \#* \langle q \cdot P' \rangle$ $\langle (bn(q \cdot \alpha)) \rangle \#* \Psi$ $\langle (bn(q \cdot \alpha)) \rangle \#* Q$ $\langle (bn(q \cdot \alpha)) \rangle \#* P$
 $\langle (bn(q \cdot \alpha)) \rangle \#* C$ $\langle p \cdot A_Q \rangle \#* C$ $\langle bn(q \cdot \alpha) \rangle \#* \text{subject}(q \cdot \alpha)$ $\langle \text{distinct}(bn(q \cdot \alpha)) \rangle$
by(*metis rPar1*)
next
case($cPar2 \Psi_P Q \alpha Q' P A_P$)
obtain $q::\text{name prm}$ **where** $\langle bn(q \cdot \alpha) \rangle \#* \Psi$ **and** $\langle bn(q \cdot \alpha) \rangle \#* P$ **and** $\langle bn(q \cdot \alpha) \rangle \#* Q$
and $\langle bn(q \cdot \alpha) \rangle \#* \alpha$ **and** $\langle bn(q \cdot \alpha) \rangle \#* A_P$ **and** $\langle bn(q \cdot \alpha) \rangle \#* Q'$ **and** $\langle bn(q \cdot \alpha) \rangle \#* \Psi_P$

and *distinctPerm* q
and $(bn(q \cdot \alpha)) \#* C$ **and** $Sq: (set\ q) \subseteq set(bn\ \alpha) \times (set(bn(q \cdot \alpha)))$
by (*rule name-list-avoiding*[**where** $xvec=bn\ \alpha$ **and** $c=(\Psi, P, Q, \alpha, A_P, \Psi_P, Q', C)$]) (*auto simp add: eqvts*)
obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#* Q$
and $(p \cdot A_P) \#* \alpha$ **and** $(p \cdot A_P) \#* (q \cdot \alpha)$ **and** $(p \cdot A_P) \#* Q'$
and $(p \cdot A_P) \#* (q \cdot Q')$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* C$
and $Sp: (set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$ **and** *distinctPerm* p
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, \alpha, q \cdot \alpha, Q', (q \cdot Q'), \Psi_P, C)$]) *auto*
from $\langle A_P \#* \alpha \rangle \langle bn(q \cdot \alpha) \#* A_P \rangle Sq \langle distinctPerm\ q \rangle$ **have** $A_P \#* (q \cdot \alpha)$
by(*subst fresh-star-bij[symmetric, of - - q]*) (*simp add: eqvts*)
from $\langle bn\ \alpha \#* subject\ \alpha \rangle \langle distinctPerm\ q \rangle$ **have** $bn(q \cdot \alpha) \#* subject(q \cdot \alpha)$
by(*subst fresh-star-bij[symmetric, of - - q]*) (*simp add: eqvts*)
from $\langle distinct(bn\ \alpha) \rangle \langle distinctPerm\ q \rangle$ **have** $distinct(bn(q \cdot \alpha))$
by(*subst distinctClosed[symmetric, of - q]*) (*simp add: eqvts*)
note $\langle cP = P \parallel Q \rangle$

moreover from $\langle cRs = \alpha \prec (P \parallel Q') \rangle \langle bn\ \alpha \#* subject\ \alpha \rangle \langle (bn(q \cdot \alpha)) \#* \alpha \rangle$
 $\langle (bn(q \cdot \alpha)) \#* Q' \rangle \langle (bn(q \cdot \alpha)) \#* P \rangle \langle bn\ \alpha \#* P \rangle Sq$
have $cRs = (q \cdot \alpha) \prec P \parallel (q \cdot Q')$
by(*force simp add: residualAlpha*)
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle \langle bn\ \alpha \#* subject\ \alpha \rangle \langle (bn(q \cdot \alpha)) \#*$
 $\alpha \rangle \langle (bn(q \cdot \alpha)) \#* Q' \rangle Sq$
have $Trans: \Psi \otimes \Psi_P \triangleright Q \mapsto (q \cdot \alpha) \prec (q \cdot Q')$
by(*force simp add: residualAlpha*)
then have $A_P \#* (q \cdot Q')$ **using** $\langle bn(q \cdot \alpha) \#* subject(q \cdot \alpha) \rangle \langle distinct(bn(q \cdot \alpha)) \rangle$
 $\langle A_P \#* Q \rangle \langle A_P \#* (q \cdot \alpha) \rangle$
by(*auto dest: freeFreshChainDerivative*)

from $Trans$ **have** $(p \cdot (\Psi \otimes \Psi_P)) \triangleright (p \cdot Q) \mapsto p \cdot ((q \cdot \alpha) \prec (q \cdot Q'))$
by(*rule semantics.eqt*)
with $\langle A_P \#* \Psi \rangle \langle A_P \#* Q \rangle \langle A_P \#* (q \cdot \alpha) \rangle \langle A_P \#* (q \cdot Q') \rangle \langle (bn(q \cdot \alpha)) \#* A_P \rangle$
 $\langle (p \cdot A_P) \#* (q \cdot \alpha) \rangle$
 $\langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (p \cdot A_P) \#* (q \cdot Q') \rangle Sp$
have $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (q \cdot \alpha) \prec (q \cdot Q')$ **by**(*simp add: eqvts*)
moreover from $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle \langle (p \cdot A_P) \#* \Psi_P \rangle Sp$ **have**
 $extractFrame\ P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha' eqvts*)
moreover from $\langle (bn(q \cdot \alpha)) \#* \Psi_P \rangle \langle (bn(q \cdot \alpha)) \#* A_P \rangle \langle (p \cdot A_P) \#* (q \cdot \alpha) \rangle$
 Sp **have** $(bn(q \cdot \alpha)) \#* (p \cdot \Psi_P)$
by(*simp add: freshAlphaPerm*)
moreover from $\langle distinct\ A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by** *simp*
ultimately show *?thesis*
using $\langle (p \cdot A_P) \#* P \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot A_P) \#* (q \cdot \alpha) \rangle$
 $\langle (p \cdot A_P) \#* (q \cdot Q') \rangle \langle (bn(q \cdot \alpha)) \#* \Psi \rangle \langle (bn(q \cdot \alpha)) \#* Q \rangle \langle (bn(q \cdot \alpha)) \#* P \rangle$
 $\langle (bn(q \cdot \alpha)) \#* C \rangle \langle (p \cdot A_P) \#* C \rangle \langle bn(q \cdot \alpha) \#* subject(q \cdot \alpha) \rangle \langle distinct(bn(q \cdot \alpha)) \rangle$

```

    by(metis rPar2)
next
  case(cComm1  $\Psi_Q$   $P$   $M$   $N$   $P'$   $A_P$   $\Psi_P$   $Q$   $K$   $xvec$   $Q'$   $A_Q$ )
  obtain  $r::name\ prm$  where  $(r \cdot xvec) \#* \Psi$  and  $(r \cdot xvec) \#* P$  and  $(r \cdot xvec)$ 
 $\#* Q$  and  $(r \cdot xvec) \#* M$ 
    and  $(r \cdot xvec) \#* K$  and  $(r \cdot xvec) \#* N$  and  $(r \cdot xvec) \#* A_P$  and  $(r \cdot xvec)$ 
 $\#* A_Q$ 
    and  $(r \cdot xvec) \#* P'$  and  $(r \cdot xvec) \#* Q'$  and  $(r \cdot xvec) \#* \Psi_P$  and  $(r \cdot xvec)$ 
 $\#* \Psi_Q$ 
    and  $(r \cdot xvec) \#* C$  and  $Sr: (set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$  and
distinctPerm r
    by(rule name-list-avoiding[where  $xvec=xvec$  and  $c=(\Psi, P, Q, M, K, N, A_P,$ 
 $A_Q, \Psi_P, \Psi_Q, P', Q', C)$ ])
      (auto simp add: eqvts)

  obtain  $q::name\ prm$  where  $(q \cdot A_Q) \#* \Psi$  and  $(q \cdot A_Q) \#* P$  and  $(q \cdot A_Q) \#*$ 
 $Q$  and  $(q \cdot A_Q) \#* K$ 
    and  $(q \cdot A_Q) \#* N$  and  $(q \cdot A_Q) \#* xvec$  and  $(q \cdot A_Q) \#* Q'$  and  $(q \cdot A_Q) \#*$ 
 $P'$ 
    and  $(q \cdot A_Q) \#* \Psi_P$  and  $(q \cdot A_Q) \#* A_P$  and  $(q \cdot A_Q) \#* \Psi_Q$  and  $(q \cdot A_Q)$ 
 $\#* (r \cdot xvec)$ 
    and  $(q \cdot A_Q) \#* C$  and  $Sq: (set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$ 
    by(rule name-list-avoiding[where  $xvec=A_Q$  and  $c=(\Psi, P, Q, K, N, xvec, r \cdot$ 
 $xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$ ]) clarsimp)

  obtain  $p::name\ prm$  where  $(p \cdot A_P) \#* \Psi$  and  $(p \cdot A_P) \#* P$  and  $(p \cdot A_P) \#*$ 
 $Q$  and  $(p \cdot A_P) \#* M$ 
    and  $(p \cdot A_P) \#* N$  and  $(p \cdot A_P) \#* xvec$  and  $(p \cdot A_P) \#* Q'$  and  $(p \cdot A_P) \#*$ 
 $A_Q$ 
    and  $(p \cdot A_P) \#* P'$  and  $(p \cdot A_P) \#* \Psi_P$  and  $(p \cdot A_P) \#* \Psi_Q$  and  $(p \cdot A_P) \#*$ 
 $(q \cdot A_Q)$ 
    and  $(p \cdot A_P) \#* C$  and  $(p \cdot A_P) \#* (r \cdot xvec)$  and  $Sp: (set\ p) \subseteq (set\ A_P) \times$ 
 $(set(p \cdot A_P))$ 
    by(rule name-list-avoiding[where  $xvec=A_P$  and  $c=(\Psi, P, Q, M, N, xvec, r \cdot$ 
 $xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$ ])
      (auto simp add: eqvts fresh-star-prod)

  have  $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$  by fact
  have  $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$  by fact

  from  $\langle A_P \#* Q \rangle$   $FrQ$   $\langle A_P \#* A_Q \rangle$  have  $A_P \#* \Psi_Q$ 
    by(auto dest: extractFrameFreshChain)
  from  $\langle A_Q \#* P \rangle$   $FrP$   $\langle A_P \#* A_Q \rangle$  have  $A_Q \#* \Psi_P$ 
    by(auto dest: extractFrameFreshChain)
  note  $\langle cP = P \parallel Q \rangle$ 
  moreover from  $\langle (r \cdot xvec) \#* P' \rangle$   $\langle (r \cdot xvec) \#* Q' \rangle$  have  $(r \cdot xvec) \#* (P' \parallel$ 
 $Q')$ 
    by simp
  with  $\langle cRs = \tau \prec (\nu*xvec)(P' \parallel Q') \rangle$   $\langle (r \cdot xvec) \#* N \rangle$   $Sr$ 

```

have $cRs = \tau \prec (\nu^*(r \cdot xvec)) \langle r \cdot (P' \parallel Q') \rangle$ **by** (*simp add: resChainAlpha residualInject*)

then have $cRs = \tau \prec (\nu^*(r \cdot xvec)) \langle (r \cdot P') \parallel (r \cdot Q') \rangle$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M \langle N \rangle \prec P' \rangle$ *Sr* $\langle \text{distinctPerm } r \rangle \langle xvec \ \#^* P \rangle \langle (r \cdot xvec) \ \#^* P \rangle$

have $\Psi \otimes \Psi_Q \triangleright P \mapsto M \langle (r \cdot N) \rangle \prec (r \cdot P')$

by (*rule inputAlpha*)

then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto (q \cdot M) \langle (r \cdot N) \rangle \prec (r \cdot P')$ **using** *Sq* $\langle A_Q \ \#^* P \rangle \langle (q \cdot A_Q) \ \#^* P \rangle$

by $-$ (*rule inputPermFrameSubject, (assumption | simp)+*)

then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto (q \cdot M) \langle (r \cdot N) \rangle \prec (r \cdot P')$ **using** *Sq* $\langle A_Q \ \#^* \Psi \rangle \langle (q \cdot A_Q) \ \#^* \Psi \rangle$

by (*simp add: eqvts*)

moreover from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ *Sp* $\langle (p \cdot A_P) \ \#^* \Psi_P \rangle$

have $FrP: \text{extractFrame } P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$

by (*simp add: frameChainAlpha*)

moreover from $\langle \text{distinct } A_P \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K \langle \nu^* xvec \rangle \langle N \rangle \prec Q' \rangle$ *Sr* $\langle (r \cdot xvec) \ \#^* N \rangle \langle (r \cdot xvec) \ \#^* Q' \rangle$

have $\Psi \otimes \Psi_P \triangleright Q \mapsto K \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot Q')$

by (*simp add: boundOutputChainAlpha'' residualInject*)

then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto (p \cdot K) \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot Q')$
using *Sp* $\langle A_P \ \#^* Q \rangle \langle (p \cdot A_P) \ \#^* Q \rangle \langle (r \cdot xvec) \ \#^* K \rangle \langle (p \cdot A_P) \ \#^* (r \cdot xvec) \rangle \langle (r \cdot xvec) \ \#^* A_P \rangle$

by (*fastforce intro: outputPermFrameSubject*)

then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (p \cdot K) \langle \nu^*(r \cdot xvec) \rangle \langle (r \cdot N) \rangle \prec (r \cdot Q')$ **using** *Sp* $\langle A_P \ \#^* \Psi \rangle \langle (p \cdot A_P) \ \#^* \Psi \rangle$

by (*simp add: eqvts*)

moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ *Sq* $\langle (q \cdot A_Q) \ \#^* \Psi_Q \rangle$

have $FrQ: \text{extractFrame } Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$

by (*simp add: frameChainAlpha*)

moreover from $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(q \cdot A_Q)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash (p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$

by (*metis chanEqClosed*)

with $\langle A_P \ \#^* \Psi \rangle \langle (p \cdot A_P) \ \#^* \Psi \rangle \langle A_Q \ \#^* \Psi \rangle \langle (q \cdot A_Q) \ \#^* \Psi \rangle \langle A_Q \ \#^* \Psi_P \rangle \langle (q \cdot A_Q) \ \#^* \Psi_P \rangle$

$\langle A_P \ \#^* \Psi_Q \rangle \langle (p \cdot A_P) \ \#^* \Psi_Q \rangle \langle A_P \ \#^* M \rangle \langle (p \cdot A_P) \ \#^* M \rangle \langle (q \cdot A_Q) \ \#^* A_P \rangle$

$\langle (p \cdot A_P) \ \#^* (q \cdot A_Q) \rangle$

$\langle A_Q \ \#^* K \rangle \langle (q \cdot A_Q) \ \#^* K \rangle \langle A_P \ \#^* A_Q \rangle \langle (p \cdot A_P) \ \#^* A_Q \rangle$ *Sp Sq*

have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$

by (*simp add: eqvts freshChainSimps*)

moreover note $\langle (p \cdot A_P) \ \#^* \Psi \rangle$

moreover from $\langle (p \cdot A_P) \ \#^* A_Q \rangle \langle (p \cdot A_P) \ \#^* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \ \#^* \Psi_Q \rangle$ *Sq*

have $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* P \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* M \rangle Sq$
have $(p \cdot A_P) \#* (q \cdot M)$
by(*simp add: freshChainSimps*)
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* N \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot N)$
by(*simp add: freshChainSimps*)
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* P' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot P')$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* Q \rangle$
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* Q' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot Q')$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle Sp Sq$ **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* P \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* K \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle Sp Sq$ **have** $(q \cdot A_Q) \#* (p \cdot K)$
by(*simp add: freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* N \rangle Sr$
have $(q \cdot A_Q) \#* (r \cdot N)$
by(*simp add: freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* P' \rangle Sr$
have $(q \cdot A_Q) \#* (r \cdot P')$
by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* Q \rangle$
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* Q' \rangle Sr$
have $(q \cdot A_Q) \#* (r \cdot Q')$
by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi \rangle$
moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_P \rangle Sp$ **have** $(r \cdot xvec) \#* (p \cdot \Psi_P)$
by(*simp add: freshChainSimps*)
moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_Q \rangle Sq$
have $(r \cdot xvec) \#* (q \cdot \Psi_Q)$
by(*simp add: freshChainSimps*)
moreover note $\langle (r \cdot xvec) \#* P \rangle$
moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* M \rangle Sq$
have $(r \cdot xvec) \#* (q \cdot M)$
by(*simp add: freshChainSimps*)
moreover note $\langle (r \cdot xvec) \#* Q \rangle$
moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* K \rangle Sp$
have $(r \cdot xvec) \#* (p \cdot K)$
by(*simp add: freshChainSimps*)

moreover note $\langle (p \cdot A_P) \#* C \rangle \langle (q \cdot A_Q) \#* C \rangle \langle (r \cdot xvec) \#* C \rangle$
moreover from $\langle distinct\ xvec \rangle$ **have** $distinct(r \cdot xvec)$ **by simp**
ultimately show $?thesis$ **by**(rule rComm1)
next
case(cComm2 Ψ_Q P M $xvec$ N P' A_P Ψ_P Q K Q' A_Q)
obtain $r::name\ prm$ **where** $(r \cdot xvec) \#* \Psi$ **and** $(r \cdot xvec) \#* P$ **and** $(r \cdot xvec)$
 $\#* Q$ **and** $(r \cdot xvec) \#* M$
and $(r \cdot xvec) \#* K$ **and** $(r \cdot xvec) \#* N$ **and** $(r \cdot xvec) \#* A_P$ **and** $(r \cdot xvec)$
 $\#* A_Q$
and $(r \cdot xvec) \#* P'$ **and** $(r \cdot xvec) \#* Q'$ **and** $(r \cdot xvec) \#* \Psi_P$ **and** $(r \cdot xvec)$
 $\#* \Psi_Q$
and $(r \cdot xvec) \#* C$ **and** $Sr: (set\ r) \subseteq (set\ xvec) \times (set(r \cdot xvec))$ **and**
 $distinctPerm\ r$
by(rule name-list-avoiding[**where** $xvec=xvec$ **and** $c=(\Psi, P, Q, M, K, N, A_P,$
 $A_Q, \Psi_P, \Psi_Q, P', Q', C)$])
 $(auto\ simp\ add: eqvts)$

obtain $q::name\ prm$ **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$ **and** $(q \cdot A_Q) \#*$
 Q **and** $(q \cdot A_Q) \#* K$
and $(q \cdot A_Q) \#* N$ **and** $(q \cdot A_Q) \#* xvec$ **and** $(q \cdot A_Q) \#* Q'$ **and** $(q \cdot A_Q) \#*$
 P'
and $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$ **and** $(q \cdot A_Q)$
 $\#* (r \cdot xvec)$
and $(q \cdot A_Q) \#* C$ **and** $Sq: (set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$
by(rule name-list-avoiding[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, K, N, xvec, r \cdot$
 $xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$]) $clarsimp$

obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#*$
 Q **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* N$ **and** $(p \cdot A_P) \#* xvec$ **and** $(p \cdot A_P) \#* Q'$ **and** $(p \cdot A_P) \#*$
 A_Q
and $(p \cdot A_P) \#* P'$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#*$
 $(q \cdot A_Q)$
and $(p \cdot A_P) \#* C$ **and** $(p \cdot A_P) \#* (r \cdot xvec)$ **and** $Sp: (set\ p) \subseteq (set\ A_P) \times$
 $(set(p \cdot A_P))$
by(rule name-list-avoiding[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, N, xvec, r \cdot$
 $xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$])
 $(auto\ simp\ add: eqvts\ fresh-star-prod)$

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by fact**
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $\langle A_P \#* Q \rangle$ FrQ $\langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(auto dest: extractFrameFreshChain)
from $\langle A_Q \#* P \rangle$ FrP $\langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(auto dest: extractFrameFreshChain)

note $\langle cP = P \parallel Q \rangle$
moreover from $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$ **have** $(r \cdot xvec) \#* (P' \parallel$

Q'
by *simp*
with $\langle cRs = \tau \prec (\nu * xvec)(P' \parallel Q') \rangle \langle (r \cdot xvec) \#* N \rangle Sr$
have $cRs = \tau \prec (\nu * (r \cdot xvec))(r \cdot (P' \parallel Q'))$ **by** (*simp add: resChainAlpha residualInject*)
then have $cRs = \tau \prec (\nu * (r \cdot xvec))((r \cdot P') \parallel (r \cdot Q'))$
by *simp*

moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)(N) \prec P' \rangle Sr \langle (r \cdot xvec) \#* N \rangle$
 $\langle (r \cdot xvec) \#* P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * (r \cdot xvec))((r \cdot N)) \prec (r \cdot P')$ **by** (*simp add: boundOutputChainAlpha'' residualInject*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto (q \cdot M)(\nu * (r \cdot xvec))((r \cdot N)) \prec (r \cdot P')$
using $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle (r \cdot xvec) \#* M \rangle \langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle$
by (*fastforce intro: outputPermFrameSubject*)
then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto (q \cdot M)(\nu * (r \cdot xvec))((r \cdot N)) \prec (r \cdot P')$ **using** $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
by (*simp add: eqvts*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by (*simp add: frameChainAlpha*)
moreover from $\langle distinct A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q' \rangle Sr \langle distinctPerm r \rangle \langle xvec \#* Q \rangle$
 $\langle (r \cdot xvec) \#* Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto K((r \cdot N)) \prec (r \cdot Q')$ **by** (*rule inputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto (p \cdot K)((r \cdot N)) \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle$
by $- (rule inputPermFrameSubject, (assumption \mid simp) +)$
then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto (p \cdot K)((r \cdot N)) \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by (*simp add: eqvts*)

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: extractFrame Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by (*simp add: frameChainAlpha*)
moreover from $\langle distinct A_Q \rangle$ **have** $distinct(q \cdot A_Q)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $(p \cdot q \cdot (\Psi \otimes \Psi_P \otimes \Psi_Q)) \vdash$
 $(p \cdot q \cdot M) \leftrightarrow (p \cdot q \cdot K)$
by (*metis chanEqClosed*)
with $\langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle \langle (q \cdot A_Q) \#* A_P \rangle$
 $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$
 $\langle A_Q \#* K \rangle \langle (q \cdot A_Q) \#* K \rangle \langle A_P \#* A_Q \rangle \langle (p \cdot A_P) \#* A_Q \rangle Sp Sq$
have $\Psi \otimes (p \cdot \Psi_P) \otimes (q \cdot \Psi_Q) \vdash (q \cdot M) \leftrightarrow (p \cdot K)$

by(*simp add: eqvts freshChainSimps*)
 moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$
 moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle$ *Sq*
 have $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (p \cdot A_P) \#* P \rangle$
 moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* M \rangle$ *Sq*
 have $(p \cdot A_P) \#* (q \cdot M)$
 by(*simp add: freshChainSimps*)
 moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* N \rangle$ *Sr*
 have $(p \cdot A_P) \#* (r \cdot N)$
 by(*simp add: freshChainSimps*)
 moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* P' \rangle$ *Sr*
 have $(p \cdot A_P) \#* (r \cdot P')$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (p \cdot A_P) \#* Q \rangle$
 moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* Q' \rangle$ *Sr*
 have $(p \cdot A_P) \#* (r \cdot Q')$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
 moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ *Sp Sq* have $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (q \cdot A_Q) \#* P \rangle$
 moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* K \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ *Sp Sq* have $(q \cdot A_Q) \#* (p \cdot K)$
 by(*simp add: freshChainSimps*)
 moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* N \rangle$ *Sr*
 have $(q \cdot A_Q) \#* (r \cdot N)$
 by(*simp add: freshChainSimps*)
 moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* P' \rangle$ *Sr*
 have $(q \cdot A_Q) \#* (r \cdot P')$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (q \cdot A_Q) \#* Q \rangle$
 moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* Q' \rangle$ *Sr*
 have $(q \cdot A_Q) \#* (r \cdot Q')$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi \rangle$
 moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_P \rangle$ *Sp* have $(r \cdot xvec) \#* (p \cdot \Psi_P)$
 by(*simp add: freshChainSimps*)
 moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_Q \rangle$ *Sq* have $(r \cdot xvec) \#* (q \cdot \Psi_Q)$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (r \cdot xvec) \#* P \rangle$
 moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* M \rangle$ *Sq* have $(r \cdot xvec) \#* (q \cdot M)$
 by(*simp add: freshChainSimps*)
 moreover note $\langle (r \cdot xvec) \#* Q \rangle$

moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* K \rangle Sp$
have $(r \cdot xvec) \#* (p \cdot K)$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* C \rangle \langle (q \cdot A_Q) \#* C \rangle \langle (r \cdot xvec) \#* C \rangle$
moreover from $\langle distinct\ xvec \rangle$ **have** $distinct(r \cdot xvec)$ **by simp**
ultimately show *?thesis* **by**(*rule rComm2*)
next
case(*cBrMerge* $\Psi_Q P M N P' A_P \Psi_P Q Q' A_Q$)
obtain $q::name\ prm$ **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$
and $(q \cdot A_Q) \#* Q$ **and** $(q \cdot A_Q) \#* M$
and $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$
and $(q \cdot A_Q) \#* N$ **and** $(q \cdot A_Q) \#* P'$ **and** $(q \cdot A_Q) \#* Q'$
and $(q \cdot A_Q) \#* C$
and $Sq: set\ q \subseteq set\ A_Q \times set\ (q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $c=(\Psi, P, N, M, P', Q', Q, \Psi_Q, A_P, \Psi_P,$
 $C)$])
(*auto simp add: eqvts fresh-star-prod emptyFresh*)
obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$
and $(p \cdot A_P) \#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* N$ **and** $(p \cdot A_P) \#* P'$ **and** $(p \cdot A_P) \#* Q'$
and $(p \cdot A_P) \#* C$
and $Sp: set\ p \subseteq set\ A_P \times set\ (p \cdot A_P)$
and $(p \cdot A_P) \#* (q \cdot A_Q)$
by(*rule name-list-avoiding*[**where** $c=(\Psi, P, N, P', Q', Q, M, A_Q, q \cdot A_Q, \Psi_Q,$
 $\Psi_P, C)$])
(*auto simp add: eqvts fresh-star-prod*)

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by fact**
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $\langle A_P \#* Q \rangle FrQ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle FrP \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)

from $Sp \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by simp
from $Sq \langle A_Q \#* M \rangle \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by simp

note $\langle cP = P \parallel Q \rangle \langle cRs = \downarrow M \parallel N \rangle \prec (P' \parallel Q')$
moreover from $\langle distinct\ A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by simp**
moreover from $\langle distinct\ A_Q \rangle$ **have** $distinct(q \cdot A_Q)$ **by simp**
moreover from $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame\ P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)

moreover from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: \text{extractFrame } Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by $(\text{simp add: frameChainAlpha})$

moreover have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(N) \prec P'$ **using** $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P' \rangle$
by $- (\text{rule brinputPermFrameSubject}, (\text{assumption} \mid \text{simp})+)$
then have $(\Psi \otimes (q \cdot \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(N) \prec P'$ **using** $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
by (simp add: eqvts)
with $\langle (q \cdot M) = M \rangle$ **have** $PTrans: (\Psi \otimes (q \cdot \Psi_Q)) \triangleright P \mapsto_i M(N) \prec P'$
by simp

moreover have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(N) \prec Q'$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle \langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q' \rangle$
by $- (\text{rule brinputPermFrameSubject}, (\text{assumption} \mid \text{simp})+)$
then have $(\Psi \otimes (p \cdot \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(N) \prec Q'$ **using** $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle$
by (simp add: eqvts)
with $\langle (p \cdot M) = M \rangle$ **have** $PTrans: (\Psi \otimes (p \cdot \Psi_P)) \triangleright Q \mapsto_i M(N) \prec Q'$
by simp

moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle Sq$
have $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by $(\text{simp add: freshChainSimps})$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle Sp Sq$ **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
by $(\text{simp add: freshChainSimps})$

moreover note
 $\langle (p \cdot A_P) \#* \Psi \rangle$
 $\langle (p \cdot A_P) \#* P \rangle \langle (p \cdot A_P) \#* N \rangle \langle (p \cdot A_P) \#* P' \rangle \langle (p \cdot A_P) \#* M \rangle$
 $\langle (p \cdot A_P) \#* Q \rangle \langle (p \cdot A_P) \#* Q' \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
 $\langle (q \cdot A_Q) \#* P \rangle \langle (q \cdot A_Q) \#* N \rangle \langle (q \cdot A_Q) \#* P' \rangle \langle (q \cdot A_Q) \#* M \rangle$
 $\langle (q \cdot A_Q) \#* Q \rangle \langle (q \cdot A_Q) \#* Q' \rangle \langle (p \cdot A_P) \#* C \rangle \langle (q \cdot A_Q) \#* C \rangle$

ultimately show $?thesis$
by $(\text{auto simp add: rBrMerge})$

next
case $(cBrComm1 \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q)$
obtain $r::\text{name prm where } (r \cdot xvec) \#* \Psi \text{ and } (r \cdot xvec) \#* P \text{ and } (r \cdot xvec) \#* Q \text{ and } (r \cdot xvec) \#* M$
and $(r \cdot xvec) \#* N \text{ and } (r \cdot xvec) \#* A_P \text{ and } (r \cdot xvec) \#* A_Q$
and $(r \cdot xvec) \#* P' \text{ and } (r \cdot xvec) \#* Q' \text{ and } (r \cdot xvec) \#* \Psi_P \text{ and } (r \cdot xvec) \#* \Psi_Q$
and $(r \cdot xvec) \#* C \text{ and } Sr: (\text{set } r) \subseteq (\text{set } xvec) \times (\text{set}(r \cdot xvec)) \text{ and } \text{distinctPerm } r$
by $(\text{rule name-list-avoiding}[\text{where } xvec=xvec \text{ and } c=(\Psi, P, Q, M, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q', C)])$
 $(\text{auto simp add: eqvts})$

obtain $q::name\ prm$ **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$ **and** $(q \cdot A_Q) \#* Q$ **and** $(q \cdot A_Q) \#* M$
and $(q \cdot A_Q) \#* N$ **and** $(q \cdot A_Q) \#* xvec$ **and** $(q \cdot A_Q) \#* Q'$ **and** $(q \cdot A_Q) \#* P'$
and $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$ **and** $(q \cdot A_Q) \#* (r \cdot xvec)$
and $(q \cdot A_Q) \#* C$ **and** $Sq: (set\ q) \subseteq (set\ A_Q) \times (set(q \cdot A_Q))$
by(*rule name-list-avoiding*[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, M, N, xvec, r \cdot xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$]) *clarsimp*

obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* N$ **and** $(p \cdot A_P) \#* xvec$ **and** $(p \cdot A_P) \#* Q'$ **and** $(p \cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* P'$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#* (q \cdot A_Q)$
and $(p \cdot A_P) \#* C$ **and** $(p \cdot A_P) \#* (r \cdot xvec)$ **and** $S_p: (set\ p) \subseteq (set\ A_P) \times (set(p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, N, xvec, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$])
(auto simp add: eqvts fresh-star-prod)

from $S_p \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by *simp*
from $S_q \langle A_Q \#* M \rangle \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by *simp*
from $S_r \langle xvec \#* M \rangle \langle (r \cdot xvec) \#* M \rangle$
have $(r \cdot M) = M$
by *simp*

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle FrQ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle FrP \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)

note $\langle cP = P \parallel Q \rangle$
moreover from $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$ **have** $(r \cdot xvec) \#* (P' \parallel Q')$
by *simp*
with $\langle cRs = \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec (P' \parallel Q') \rangle \langle (r \cdot xvec) \#* N \rangle \langle (r \cdot xvec) \#* P' \rangle$
 $\langle (r \cdot xvec) \#* Q' \rangle \langle xvec \#* M \rangle \langle (r \cdot xvec) \#* M \rangle Sr$
have $cRs = (r \cdot (\downarrow M(\downarrow \nu * xvec) \langle N \rangle)) \prec (r \cdot (P' \parallel Q'))$
by (*simp add: residualAlpha*)
with $\langle (r \cdot M) = M \rangle$ **have** $cRs = \downarrow M(\downarrow \nu * (r \cdot xvec)) \langle (r \cdot N) \rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$

$Q')$) by *simp*

moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P' \rangle Sr \langle distinctPerm r \rangle \langle xvec \#* P \rangle \langle (r \cdot xvec) \#* P \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|(r \cdot N)|) \prec (r \cdot P')$
by (*rule brinputAlpha*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i (q \cdot M)(|(r \cdot N)|) \prec (r \cdot P')$ **using** $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle$
by $-$ (*rule brinputPermFrameSubject, (assumption | simp)+*)
then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i M(|(r \cdot N)|) \prec (r \cdot P')$ **using** $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle \langle (q \cdot M) = M \rangle$
by (*simp add: eqvts*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by (*simp add: frameChainAlpha*)
moreover from $\langle distinct A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(|\nu * xvec|)(|N|) \prec Q' \rangle Sr \langle (r \cdot xvec) \#* N \rangle \langle (r \cdot xvec) \#* Q' \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(|\nu * (r \cdot xvec)|)(|(r \cdot N)|) \prec (r \cdot Q')$
by (*simp add: boundOutputChainAlpha'' residualInject*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)(|\nu * (r \cdot xvec)|)(|(r \cdot N)|) \prec (r \cdot Q')$
using $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle \langle (r \cdot xvec) \#* M \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* A_P \rangle$
by (*fastforce intro: broutputPermFrameSubject*)
then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M(|\nu * (r \cdot xvec)|)(|(r \cdot N)|) \prec (r \cdot Q')$
using $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot M) = M \rangle$
by (*simp add: eqvts*)

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: extractFrame Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by (*simp add: frameChainAlpha*)
moreover from $\langle distinct A_Q \rangle$ **have** $distinct(q \cdot A_Q)$ **by** *simp*

moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle Sq$
have $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by (*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* P \rangle$
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* N \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot N)$
by (*simp add: freshChainSimps*)
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* P' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot P')$
by (*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* Q \rangle$
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* Q' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot Q')$

by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle$ *Sp Sq* **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
 by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* P \rangle$
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* N \rangle$ *Sr*
have $(q \cdot A_Q) \#* (r \cdot N)$
 by(*simp add: freshChainSimps*)
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* P' \rangle$ *Sr*
have $(q \cdot A_Q) \#* (r \cdot P')$
 by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* Q \rangle$
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* Q' \rangle$ *Sr*
have $(q \cdot A_Q) \#* (r \cdot Q')$
 by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi \rangle$
moreover from $\langle (r \cdot xvec) \#* A_P \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_P \rangle$
Sp **have** $(r \cdot xvec) \#* (p \cdot \Psi_P)$
 by(*simp add: freshChainSimps*)
moreover from $\langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (r \cdot xvec) \#* \Psi_Q \rangle$
Sq **have** $(r \cdot xvec) \#* (q \cdot \Psi_Q)$
 by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* M \rangle \langle (q \cdot A_Q) \#* M \rangle$
moreover note $\langle (r \cdot xvec) \#* P \rangle$
moreover note $\langle (r \cdot xvec) \#* Q \rangle \langle (r \cdot xvec) \#* M \rangle$
moreover note $\langle (p \cdot A_P) \#* C \rangle \langle (q \cdot A_Q) \#* C \rangle \langle (r \cdot xvec) \#* C \rangle$
moreover from $\langle distinct\ xvec \rangle$ **have** $distinct(r \cdot xvec)$ **by** *simp*
ultimately show *?thesis* **by** (*simp add: rBrComm1*)
next
case (*cBrComm2* Ψ_Q *P M xvec N P' A_P* Ψ_P *Q Q' A_Q*)
obtain *r::name prm* **where** $(r \cdot xvec) \#* \Psi$ **and** $(r \cdot xvec) \#* P$ **and** $(r \cdot xvec) \#* Q$ **and** $(r \cdot xvec) \#* M$
and $(r \cdot xvec) \#* N$ **and** $(r \cdot xvec) \#* A_P$ **and** $(r \cdot xvec) \#* A_Q$
and $(r \cdot xvec) \#* P'$ **and** $(r \cdot xvec) \#* Q'$ **and** $(r \cdot xvec) \#* \Psi_P$ **and** $(r \cdot xvec) \#* \Psi_Q$
and $(r \cdot xvec) \#* C$ **and** *Sr: (set r) \subseteq (set xvec) \times (set(r \cdot xvec))* **and**
distinctPerm r
by (*rule name-list-avoiding[where xvec=xvec and c=($\Psi, P, Q, M, N, A_P, A_Q, \Psi_P, \Psi_Q, P', Q', C$)]*)
(auto simp add: eqvts)
obtain *q::name prm* **where** $(q \cdot A_Q) \#* \Psi$ **and** $(q \cdot A_Q) \#* P$ **and** $(q \cdot A_Q) \#* Q$ **and** $(q \cdot A_Q) \#* M$
and $(q \cdot A_Q) \#* N$ **and** $(q \cdot A_Q) \#* xvec$ **and** $(q \cdot A_Q) \#* Q'$ **and** $(q \cdot A_Q) \#* P'$
and $(q \cdot A_Q) \#* \Psi_P$ **and** $(q \cdot A_Q) \#* A_P$ **and** $(q \cdot A_Q) \#* \Psi_Q$ **and** $(q \cdot A_Q) \#* (r \cdot xvec)$
and $(q \cdot A_Q) \#* C$ **and** *Sq: (set q) \subseteq (set A_Q) \times (set(q \cdot A_Q))*

by(*rule name-list-avoiding*[**where** $xvec=A_Q$ **and** $c=(\Psi, P, Q, N, M, xvec, r \cdot xvec, \Psi_Q, A_P, \Psi_P, Q', P', C)$]) *clarsimp*

obtain $p::name\ prm$ **where** $(p \cdot A_P) \#* \Psi$ **and** $(p \cdot A_P) \#* P$ **and** $(p \cdot A_P) \#* Q$ **and** $(p \cdot A_P) \#* M$
and $(p \cdot A_P) \#* N$ **and** $(p \cdot A_P) \#* xvec$ **and** $(p \cdot A_P) \#* Q'$ **and** $(p \cdot A_P) \#* A_Q$
and $(p \cdot A_P) \#* P'$ **and** $(p \cdot A_P) \#* \Psi_P$ **and** $(p \cdot A_P) \#* \Psi_Q$ **and** $(p \cdot A_P) \#* (q \cdot A_Q)$
and $(p \cdot A_P) \#* C$ **and** $(p \cdot A_P) \#* (r \cdot xvec)$ **and** $S_p: (set\ p) \subseteq (set\ A_P) \times (set\ (p \cdot A_P))$
by(*rule name-list-avoiding*[**where** $xvec=A_P$ **and** $c=(\Psi, P, Q, M, N, xvec, r \cdot xvec, A_Q, q \cdot A_Q, \Psi_Q, \Psi_P, Q', P', C)$])
(auto simp add: eqvts fresh-star-prod)

from $S_p \langle A_P \#* M \rangle \langle (p \cdot A_P) \#* M \rangle$
have $(p \cdot M) = M$
by *simp*
from $S_q \langle A_Q \#* M \rangle \langle (q \cdot A_Q) \#* M \rangle$
have $(q \cdot M) = M$
by *simp*
from $S_r \langle xvec \#* M \rangle \langle (r \cdot xvec) \#* M \rangle$
have $(r \cdot M) = M$
by *simp*

have $FrP: extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have $FrQ: extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle FrQ \langle A_P \#* A_Q \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle A_Q \#* P \rangle FrP \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)

note $\langle cP = P \parallel Q \rangle$
moreover from $\langle (r \cdot xvec) \#* P' \rangle \langle (r \cdot xvec) \#* Q' \rangle$ **have** $(r \cdot xvec) \#* (P' \parallel Q')$
by *simp*
with $\langle cRs = {}_iM(\nu*xvec)\langle N \rangle \prec (P' \parallel Q') \rangle \langle (r \cdot xvec) \#* N \rangle \langle (r \cdot xvec) \#* P' \rangle$
 $\langle (r \cdot xvec) \#* Q' \rangle \langle xvec \#* M \rangle \langle (r \cdot xvec) \#* M \rangle Sr$
have $cRs = (r \cdot ({}_iM(\nu*xvec)\langle N \rangle)) \prec (r \cdot (P' \parallel Q'))$
by (*simp add: residualAlpha*)
with $\langle (r \cdot M) = M \rangle$ **have** $cRs = {}_iM(\nu*(r \cdot xvec))\langle (r \cdot N) \rangle \prec ((r \cdot P') \parallel (r \cdot Q'))$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto {}_iM(\nu*xvec)\langle N \rangle \prec P' \rangle Sr \langle (r \cdot xvec) \#* N \rangle$
 $\langle (r \cdot xvec) \#* P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto {}_iM(\nu*(r \cdot xvec))\langle (r \cdot N) \rangle \prec (r \cdot P')$ **by**(*simp add: boundOutputChainAlpha'' residualInject*)
then have $(q \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto {}_i(q \cdot M)(\nu*(r \cdot xvec))\langle (r \cdot N) \rangle \prec (r \cdot P')$

using $Sq \langle A_Q \#* P \rangle \langle (q \cdot A_Q) \#* P \rangle \langle (r \cdot xvec) \#* M \rangle \langle (r \cdot xvec) \#* A_Q \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle$
by(*fastforce intro: brouputPermFrameSubject*)
then have $PTrans: \Psi \otimes (q \cdot \Psi_Q) \triangleright P \mapsto_i M(\nu^*(r \cdot xvec)) \langle (r \cdot N) \rangle \prec (r \cdot P')$
using $Sq \langle A_Q \#* \Psi \rangle \langle (q \cdot A_Q) \#* \Psi \rangle \langle (q \cdot M) = M \rangle$
by(*simp add: eqvts*)

moreover from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle Sp \langle (p \cdot A_P) \#* \Psi_P \rangle$
have $FrP: extractFrame P = \langle (p \cdot A_P), (p \cdot \Psi_P) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle distinct A_P \rangle$ **have** $distinct(p \cdot A_P)$ **by** *simp*

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q' \rangle Sr \langle distinctPerm r \rangle \langle xvec \#* Q \rangle \langle (r \cdot xvec) \#* Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M((r \cdot N)) \prec (r \cdot Q')$ **by**(*rule brinputAlpha*)
then have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i (p \cdot M)((r \cdot N)) \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* Q \rangle \langle (p \cdot A_P) \#* Q \rangle$
by $- (rule brinputPermFrameSubject, (assumption \mid simp)+)$
then have $QTrans: \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M((r \cdot N)) \prec (r \cdot Q')$ **using** $Sp \langle A_P \#* \Psi \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot M) = M \rangle$
by(*simp add: eqvts*)

moreover from $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Sq \langle (q \cdot A_Q) \#* \Psi_Q \rangle$
have $FrQ: extractFrame Q = \langle (q \cdot A_Q), (q \cdot \Psi_Q) \rangle$
by(*simp add: frameChainAlpha*)
moreover from $\langle distinct A_Q \rangle$ **have** $distinct(q \cdot A_Q)$ **by** *simp*

moreover note $\langle (p \cdot A_P) \#* \Psi \rangle$
moreover from $\langle (p \cdot A_P) \#* A_Q \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* \Psi_Q \rangle Sq$
have $(p \cdot A_P) \#* (q \cdot \Psi_Q)$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* P \rangle$
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* N \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot N)$
by(*simp add: freshChainSimps*)
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* P' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot P')$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* Q \rangle$
moreover from $\langle (p \cdot A_P) \#* xvec \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (p \cdot A_P) \#* Q' \rangle Sr$
have $(p \cdot A_P) \#* (r \cdot Q')$
by(*simp add: freshChainSimps*)
moreover note $\langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle \langle (p \cdot A_P) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* \Psi \rangle$
moreover from $\langle (q \cdot A_Q) \#* A_P \rangle \langle (p \cdot A_P) \#* A_Q \rangle \langle (q \cdot A_Q) \#* \Psi_P \rangle \langle (p \cdot A_P) \#* (q \cdot A_Q) \rangle Sp Sq$ **have** $(q \cdot A_Q) \#* (p \cdot \Psi_P)$
by(*simp add: freshChainSimps*)
moreover note $\langle (q \cdot A_Q) \#* P \rangle$
moreover from $\langle (q \cdot A_Q) \#* xvec \rangle \langle (q \cdot A_Q) \#* (r \cdot xvec) \rangle \langle (q \cdot A_Q) \#* N \rangle Sr$
have $(q \cdot A_Q) \#* (r \cdot N)$

```

    by(simp add: freshChainSimps)
  moreover from ⟨(q · AQ) #* xvec⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(q · AQ) #* P'⟩ Sr
  have (q · AQ) #* (r · P')
    by(simp add: freshChainSimps)
  moreover note ⟨(q · AQ) #* Q⟩
  moreover from ⟨(q · AQ) #* xvec⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(q · AQ) #* Q'⟩ Sr
  have (q · AQ) #* (r · Q')
    by(simp add: freshChainSimps)
  moreover note ⟨(q · AQ) #* (r · xvec)⟩ ⟨(r · xvec) #* Ψ⟩
  moreover from ⟨(r · xvec) #* AP⟩ ⟨(p · AP) #* (r · xvec)⟩ ⟨(r · xvec) #* ΨP⟩
  Sp have (r · xvec) #* (p · ΨP)
    by(simp add: freshChainSimps)
  moreover from ⟨(r · xvec) #* AQ⟩ ⟨(q · AQ) #* (r · xvec)⟩ ⟨(r · xvec) #* ΨQ⟩
  Sq have (r · xvec) #* (q · ΨQ)
    by(simp add: freshChainSimps)
  moreover note ⟨(p · AP) #* M⟩ ⟨(q · AQ) #* M⟩
  moreover note ⟨(r · xvec) #* P⟩
  moreover note ⟨(r · xvec) #* Q⟩ ⟨(r · xvec) #* M⟩
  moreover note ⟨(p · AP) #* C⟩ ⟨(q · AQ) #* C⟩ ⟨(r · xvec) #* C⟩
  moreover from ⟨distinct xvec⟩ have distinct(r · xvec) by simp
  ultimately show ?thesis by(simp add: rBrComm2)
next
  case(cBrClose P M xvec N P' x)
  obtain r::name prm where (r · xvec) #* Ψ and (r · xvec) #* P and (r · xvec)
#* M
    and (r · xvec) #* N and (r · xvec) #* P' and (r · xvec) #* x
    and (r · xvec) #* C and Sr: (set r) ⊆ (set xvec) × (set(r · xvec)) and
distinctPerm r
  by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, P, M, N, P', x, C)])
  (auto simp add: eqts)
  obtain y::name where y #* P and y #* C and y #* xvec and y ≠ x and y #* N
    and y #* (r · xvec) and y #* r and y #* M and y #* Ψ
    and y #* P' and y #* (r · P') and y #* (r · N)
  by(generate-fresh name) (auto simp add: freshChainSimps)
  from ⟨cP = (νx)P⟩ ⟨y #* P⟩ have cP-perm: cP = (νy)(([x, y]) · P) by(simp
add: alphaRes)
  from ⟨cRs = τ < (νx)((ν*xvec)P')⟩ ⟨(r · xvec) #* P'⟩ Sr
  have cRs = τ < (νx)((ν*(r · xvec))(r · P')) by(simp add: resChainAlpha)
  moreover from ⟨y #* (r · P')⟩ have y #* ((ν*(r · xvec))(r · P')) by(simp add:
resChainFresh)
  ultimately have cRs = τ < (νy)(([x, y]) · ((ν*(r · xvec))(r · P'))) by(simp
add: alphaRes)
  with ⟨(r · xvec) #* x⟩ ⟨y #* (r · xvec)⟩
  have cRs-perm: cRs = τ < (νy)((ν*(r · xvec))([x, y]) · (r · P')) by(simp add:
eqts)

  from ⟨x #* xvec⟩ ⟨(r · xvec) #* x⟩ Sr have r · x = x by simp

  from ⟨Ψ ▷ P ⟶ jM(ν*xvec)⟨N⟩ < P'⟩ ⟨(r · xvec) #* N⟩ ⟨(r · xvec) #* P'⟩

```

```

  ⟨set r ⊆ set xvec × set (r · xvec)⟩
  have Ψ ▷ P ⟶ iM(ν*(r · xvec))⟨(r · N)⟩ < (r · P')
  by(simp add: boundOutputChainAlpha'' create-residual.simps)
  then have [(x, y)] · (Ψ ▷ P ⟶ iM(ν*(r · xvec))⟨(r · N)⟩ < (r · P'))
  by(simp add: perm-bool)
  with ⟨x # Ψ⟩ ⟨y # Ψ⟩ ⟨(r · xvec) #* x⟩ ⟨y # (r · xvec)⟩
  ⟨y # (r · N)⟩
  have trans-perm: Ψ ▷ ([x, y]) · P ⟶ i([x, y]) · M(ν*(r · xvec))⟨([x, y]) ·
  (r · N)⟩ < ([x, y]) · (r · P')
  by(auto simp add: eqvts)

  note cP-perm cRs-perm
  moreover from ⟨x ∈ supp M⟩ have y ∈ supp ([x, y]) · M
  by (metis fresh-bij fresh-def swap-simps)
  moreover note trans-perm
  moreover from ⟨distinct xvec⟩ ⟨distinctPerm r⟩ have distinct (r · xvec)
  by simp
  moreover note ⟨(r · xvec) #* Ψ⟩
  moreover from ⟨(r · xvec) #* x⟩ ⟨y # (r · xvec)⟩ ⟨(r · xvec) #* P⟩
  have (r · xvec) #* ([x, y]) · P by simp
  moreover from ⟨(r · xvec) #* x⟩ ⟨y # (r · xvec)⟩ ⟨(r · xvec) #* M⟩
  have (r · xvec) #* ([x, y]) · M by simp
  moreover note ⟨y # Ψ⟩ ⟨y # (r · xvec)⟩
  ⟨(r · xvec) #* C⟩ ⟨y # C⟩
  ultimately show ?thesis
  by(rule rBrClose)
next
  case(cOpen P M xvec yvec N P' x)
  from ⟨Ψ ▷ P ⟶ M(ν*(xvec@yvec))⟨N⟩ < P'⟩ have distinct(xvec@yvec)
  by(force dest: boundOutputDistinct)
  then have xvec #* yvec by(induct xvec) auto
  obtain p where (p · yvec) #* Ψ and (p · yvec) #* P and (p · yvec) #* M
  and (p · yvec) #* yvec and (p · yvec) #* N and (p · yvec) #* P'
  and x # (p · yvec) and (p · yvec) #* xvec
  and (p · yvec) #* C and Sp: (set p) ⊆ (set yvec) × (set(p · yvec))
  by(rule name-list-avoiding[where xvec=yvec and c=(Ψ, P, M, xvec, yvec, N,
  P', x, C)])
  (auto simp add: eqvts fresh-star-prod)
  obtain q where (q · xvec) #* Ψ and (q · xvec) #* P and (q · xvec) #* M
  and (q · xvec) #* xvec and (q · xvec) #* N and (q · xvec) #* P'
  and x # (q · xvec) and (q · xvec) #* yvec
  and (q · xvec) #* p and (q · xvec) #* (p · yvec)
  and (q · xvec) #* C and Sq: (set q) ⊆ (set xvec) × (set(q · xvec))
  by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, P, M, xvec, yvec, p ·
  yvec, N, P', x, p, C)])
  (auto simp add: eqvts fresh-star-prod)
  obtain y::name where y # P and y # C and y # xvec and y # yvec and y ≠ x
  and y # N
  and y # (q · xvec) and y # (p · yvec) and y # M and y # Ψ and y # P'

```



```

    by(generate-fresh name) (auto simp add: freshChainSimps)
  from  $\langle cP = (\nu x)P \rangle \langle y \# P \rangle$  have  $cP = (\nu y)((x, y) \cdot P)$  by(simp add: alphaRes)
  moreover have  $cRs = M(\nu*((q \cdot xvec)@y\#(p \cdot yvec)))(((q@(x, y)\#p) \cdot N)) \prec$ 
   $((q@(x, y)\#p) \cdot P')$ 
  proof -
    note  $\langle cRs = M(\nu*(xvec@x\#yvec))(N) \prec P' \rangle$ 
    moreover have  $(\nu*(xvec@x\#yvec))N \prec' P' = (\nu*xvec)((\nu x)((\nu*yvec)N \prec'$ 
 $P'))$  by(simp add: boundOutputApp)
    moreover from  $\langle (p \cdot yvec) \#* N \rangle \langle (p \cdot yvec) \#* P' \rangle$  Sp have  $\dots = (\nu*xvec)((\nu x)((\nu*(p$ 
 $\cdot yvec))(p \cdot N) \prec' (p \cdot P'))$ 
    by(simp add: boundOutputChainAlpha')
    moreover with  $\langle y \# N \rangle \langle y \# P' \rangle \langle y \# (p \cdot yvec) \rangle \langle y \# yvec \rangle \langle x \# yvec \rangle \langle x \# (p \cdot$ 
 $yvec) \rangle$  Sp
    moreover have  $\dots = (\nu*xvec)((\nu y)((\nu*(p \cdot yvec))(((x, y) \cdot p \cdot N) \prec' ((x,$ 
 $y) \cdot p \cdot P'))))$ 
    by(subst alphaBoundOutput[where y=y]) (simp add: freshChainSimps eqvts)+
    moreover then have  $\dots = (\nu*xvec)((\nu y)((\nu*(p \cdot yvec))(((x, y)\#p) \cdot N) \prec'$ 
 $((x, y)\#p) \cdot P'))$ 
    by simp
    moreover from  $\langle (q \cdot xvec) \#* N \rangle \langle (q \cdot xvec) \#* P' \rangle \langle xvec \#* yvec \rangle \langle (p \cdot yvec)$ 
 $\#* xvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle$ 
     $\langle y \# xvec \rangle \langle y \# (q \cdot xvec) \rangle \langle x \# xvec \rangle \langle x \# (q \cdot xvec) \rangle$  Sp Sq
    have  $\dots = (\nu*(q \cdot xvec))((\nu y)((\nu*(p \cdot yvec))((q \cdot ((x, y)\#p) \cdot N) \prec' (q \cdot ((x,$ 
 $y)\#p) \cdot P'))))$ 
    apply(subst boundOutputChainAlpha[where p=q and xvec=xvec and yvec=yvec])
    defer
    apply assumption
    apply simp
    apply(simp add: eqvts)
    apply(simp add: eqvts)
    apply(simp add: boundOutputFreshSet(4))
    apply(rule conjI)
    apply(simp add: freshChainSimps)
    apply(simp add: freshChainSimps)
    done
    moreover then have  $\dots = (\nu*(q \cdot xvec@y\#(p \cdot yvec)))(q@(x, y)\#p) \cdot N$ 
 $\prec' ((q@(x, y)\#p) \cdot P')$ 
    by(simp only: pt2[OF pt-name-inst] boundOutputApp BOresChain.simps)
    ultimately show ?thesis
    by(simp only: residualInject)
  qed
  moreover have  $\Psi \triangleright ((x, y) \cdot P) \mapsto M(\nu*((q \cdot xvec)@y\#(p \cdot yvec)))(q@(x,$ 
 $y)\#p) \cdot N) \prec ((q@(x, y)\#p) \cdot P')$ 
  proof -
    note  $\langle \Psi \triangleright P \mapsto M(\nu*(xvec@yvec))(N) \prec P' \rangle$ 
    moreover from  $\langle (p \cdot yvec) \#* N \rangle \langle (q \cdot xvec) \#* N \rangle \langle xvec \#* yvec \rangle \langle (q \cdot xvec)$ 
 $\#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$  Sp Sq
    have  $((q@p) \cdot (xvec @ yvec)) \#* N$  apply(simp only: eqvts) apply(simp only:
    pt2[OF pt-name-inst])

```

by *simp*
moreover from $\langle (p \cdot yvec) \#* P' \rangle \langle (q \cdot xvec) \#* P' \rangle \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle Sp Sq$
have $((q@p) \cdot (xvec @ yvec)) \#* P'$ **by** (*simp del: freshAlphaPerm add: eqvts pt2[OF pt-name-inst]*)
moreover from $Sp Sq \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
have $Spq: set(q@p) \subseteq set(xvec@yvec) \times set((q@p) \cdot (xvec@yvec))$
by (*simp add: pt2[OF pt-name-inst] eqvts blast*)
ultimately have $\Psi \triangleright P \mapsto M(\nu*((q@p) \cdot (xvec@yvec))) \langle ((q@p) \cdot N) \rangle \prec ((q@p) \cdot P')$
apply (*simp only: residualInject*)
by (*erule rev-mp*) (*subst boundOutputChainAlpha, auto*)
with $Sp Sq \langle xvec \#* yvec \rangle \langle (q \cdot xvec) \#* yvec \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle \langle (p \cdot yvec) \#* xvec \rangle$
have $\Psi \triangleright P \mapsto M(\nu*((q \cdot xvec)@(p \cdot yvec))) \langle ((q@p) \cdot N) \rangle \prec ((q@p) \cdot P')$
by (*simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm*)
then have $[(x, y) \cdot \Psi] \triangleright [(x, y) \cdot P] \mapsto [(x, y) \cdot (M(\nu*((q \cdot xvec)@(p \cdot yvec)))) \langle ((q@p) \cdot N) \rangle \prec ((q@p) \cdot P')$
by (*rule semantics.eqvt*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle \langle x \# (q \cdot xvec) \rangle \langle y \# (q \cdot xvec) \rangle \langle x \# yvec \rangle \langle y \# yvec \rangle \langle x \# (p \cdot yvec) \rangle \langle y \# (p \cdot yvec) \rangle Sp Sq$
show *?thesis*
apply (*simp add: eqvts pt2[OF pt-name-inst]*)
by (*subst perm-compose[of q], simp*)+
qed
moreover from $\langle x \in supp N \rangle$ **have** $((q@(x, y)\#p) \cdot x) \in ((q@(x, y)\#p) \cdot (supp N))$
by (*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle y \# xvec \rangle \langle y \# (q \cdot xvec) \rangle Sp Sq$
have $y \in supp((q@(x, y)\#p) \cdot N)$ **by** (*simp add: pt2[OF pt-name-inst] calc-atm eqvts*)
moreover from $\langle distinct xvec \rangle$ **have** $distinct(q \cdot xvec)$ **by** *simp*
moreover from $\langle distinct yvec \rangle$ **have** $distinct(p \cdot yvec)$ **by** *simp*
moreover note $\langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle x \# M \rangle \langle x \# \Psi \rangle$
 $\langle (q \cdot xvec) \#* \Psi \rangle \langle (q \cdot xvec) \#* P \rangle \langle (q \cdot xvec) \#* M \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle$
 $\langle (p \cdot yvec) \#* \Psi \rangle \langle (p \cdot yvec) \#* P \rangle \langle (p \cdot yvec) \#* M \rangle \langle y \# (q \cdot xvec) \rangle \langle y \# (p \cdot yvec) \rangle \langle y \# M \rangle \langle y \# C \rangle \langle y \# \Psi \rangle$
 $\langle (p \cdot yvec) \#* C \rangle \langle (q \cdot xvec) \#* C \rangle$
ultimately show *Prop* **by** $- (rule rOpen, (assumption | simp)+)$
next
case (*cBrOpen P M xvec yvec N P' x*)
from $\langle \Psi \triangleright P \mapsto M(\nu*(xvec@yvec)) \langle N \rangle \prec P' \rangle$ **have** $distinct(xvec@yvec)$
by (*force dest: boundOutputDistinct*)
then have $xvec \#* yvec$ **by** (*induct xvec auto*)
obtain p **where** $(p \cdot yvec) \#* \Psi$ **and** $(p \cdot yvec) \#* P$ **and** $(p \cdot yvec) \#* M$
and $(p \cdot yvec) \#* yvec$ **and** $(p \cdot yvec) \#* N$ **and** $(p \cdot yvec) \#* P'$
and $x \# (p \cdot yvec)$ **and** $(p \cdot yvec) \#* xvec$

```

and (p · yvec) #* C and Sp: (set p) ⊆ (set yvec) × (set(p · yvec))
by(rule name-list-avoiding[where xvec=yvec and c=(Ψ, P, M, xvec, yvec, N,
P', x, C)])
  (auto simp add: eqvts fresh-star-prod)
obtain q where (q · xvec) #* Ψ and (q · xvec) #* P and (q · xvec) #* M
and (q · xvec) #* xvec and (q · xvec) #* N and (q · xvec) #* P'
and x # (q · xvec) and (q · xvec) #* yvec
and (q · xvec) #* p and (q · xvec) #* (p · yvec)
and (q · xvec) #* C and Sq: (set q) ⊆ (set xvec) × (set(q · xvec))
by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, P, M, xvec, yvec, p ·
yvec, N, P', x, p, C)])
  (auto simp add: eqvts fresh-star-prod)
obtain y::name where y # P and y # C and y # xvec and y # yvec and y ≠ x
and y # N
and y # (q · xvec) and y # (p · yvec) and y # M and y # Ψ and y # P'
by(generate-fresh name) (auto simp add: freshChainSimps)
from ⟨cP = (νx)P⟩ ⟨y # P⟩ have cP = (νy)(([x, y] · P) by(simp add: alphaRes)
moreover have cRs = ;M(ν*(q · xvec)@y#(p · yvec))(((q@(x, y)#p) · N))
< ((q@(x, y)#p) · P')
proof –
  note ⟨cRs = ;M(ν*(xvec@x#yvec))⟨N⟩ < P'⟩
  moreover have (ν*(xvec@x#yvec))N <' P' = (ν*xvec)((νx)((ν*yvec)N <'
P')) by(simp add: boundOutputApp)
  moreover from ⟨p · yvec⟩ #* N⟩ ⟨p · yvec⟩ #* P'⟩ Sp have ... = (ν*xvec)((νx)((ν*(p
· yvec))⟨p · N⟩ <' (p · P')))
    by(simp add: boundOutputChainAlpha'')
  moreover with ⟨y # N⟩ ⟨y # P'⟩ ⟨y # (p · yvec)⟩ ⟨y # yvec⟩ ⟨x # yvec⟩ ⟨x # (p ·
yvec)⟩ Sp
  moreover have ... = (ν*xvec)((νy)((ν*(p · yvec))(((x, y) · p · N) <' ((x,
y) · p · P'))))
    by(subst alphaBoundOutput[where y=y]) (simp add: freshChainSimps eqvts)+
  moreover then have ... = (ν*xvec)((νy)((ν*(p · yvec))(((x, y)#p) · N) <'
(((x, y)#p) · P'))))
    by simp
  moreover from ⟨(q · xvec) #* N⟩ ⟨(q · xvec) #* P'⟩ ⟨xvec #* yvec⟩ ⟨p · yvec⟩
#* xvec⟩ ⟨(q · xvec) #* yvec⟩ ⟨(q · xvec) #* (p · yvec)⟩
    ⟨y # xvec⟩ ⟨y # (q · xvec)⟩ ⟨x # xvec⟩ ⟨x # (q · xvec)⟩ Sp Sq
  have ... = (ν*(q · xvec))((νy)((ν*(p · yvec))((q · ((x, y)#p) · N) <' (q · ((x,
y)#p) · P'))))
  apply(subst boundOutputChainAlpha[where p=q and xvec=xvec and yvec=yvec])
  defer
  apply assumption
  apply simp
  apply(simp add: eqvts)
  apply(simp add: eqvts)
  apply(simp add: boundOutputFreshSet(4))
  apply(rule conjI)
  apply(simp add: freshChainSimps)
  apply(simp add: freshChainSimps)

```

done
moreover then have ... = $(\nu^*(q \cdot xvec @ y \# (p \cdot yvec))) \langle (q @ (x, y) \# p) \cdot N \rangle$
 $\prec' ((q @ (x, y) \# p) \cdot P')$
by(*simp only: pt2[OF pt-name-inst] boundOutputApp BOresChain.simps*)
ultimately show *?thesis*
by(*simp only: residualInject*)
qed
moreover have $\Psi \triangleright ((x, y) \cdot P) \mapsto_{iM} (\nu^*((q \cdot xvec) @ (p \cdot yvec))) \langle (q @ (x, y) \# p) \cdot N \rangle \prec ((q @ (x, y) \# p) \cdot P')$
proof –
note $\langle \Psi \triangleright P \mapsto_{iM} (\nu^*(xvec @ yvec)) \rangle \langle N \rangle \prec P'$
moreover from $\langle (p \cdot yvec) \# N \rangle \langle (q \cdot xvec) \# N \rangle \langle xvec \# yvec \rangle \langle (q \cdot xvec) \# yvec \rangle \langle (q \cdot xvec) \# (p \cdot yvec) \rangle \langle (p \cdot yvec) \# xvec \rangle Sp Sq$
have $((q @ p) \cdot (xvec @ yvec)) \# N$ **apply**(*simp only: eqvts*) **apply**(*simp only: pt2[OF pt-name-inst]*)
by *simp*
moreover from $\langle (p \cdot yvec) \# P' \rangle \langle (q \cdot xvec) \# P' \rangle \langle xvec \# yvec \rangle \langle (q \cdot xvec) \# yvec \rangle \langle (q \cdot xvec) \# (p \cdot yvec) \rangle \langle (p \cdot yvec) \# xvec \rangle Sp Sq$
have $((q @ p) \cdot (xvec @ yvec)) \# P'$ **by**(*simp del: freshAlphaPerm add: eqvts pt2[OF pt-name-inst]*)
moreover from $Sp Sq \langle xvec \# yvec \rangle \langle (q \cdot xvec) \# yvec \rangle \langle (q \cdot xvec) \# (p \cdot yvec) \rangle \langle (p \cdot yvec) \# xvec \rangle$
have $Spq: set(q @ p) \subseteq set(xvec @ yvec) \times set((q @ p) \cdot (xvec @ yvec))$
by(*simp add: pt2[OF pt-name-inst] eqvts blast*)
ultimately have $\Psi \triangleright P \mapsto_{iM} (\nu^*((q @ p) \cdot (xvec @ yvec))) \langle (q @ p) \cdot N \rangle \prec ((q @ p) \cdot P')$
apply(*simp only: residualInject*)
by(*erule rev-mp*) (*subst boundOutputChainAlpha, auto*)
with $Sp Sq \langle xvec \# yvec \rangle \langle (q \cdot xvec) \# yvec \rangle \langle (q \cdot xvec) \# (p \cdot yvec) \rangle \langle (p \cdot yvec) \# xvec \rangle$
have $\Psi \triangleright P \mapsto_{iM} (\nu^*((q \cdot xvec) @ (p \cdot yvec))) \langle (q @ p) \cdot N \rangle \prec ((q @ p) \cdot P')$
by(*simp add: eqvts pt2[OF pt-name-inst] del: freshAlphaPerm*)
then have $((x, y) \cdot \Psi) \triangleright ((x, y) \cdot P) \mapsto [(x, y)] \cdot (iM(\nu^*((q \cdot xvec) @ (p \cdot yvec)))) \langle (q @ p) \cdot N \rangle \prec ((q @ p) \cdot P')$
by(*rule semantics.eqv*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle \langle x \# (q \cdot xvec) \rangle \langle y \# (q \cdot xvec) \rangle \langle x \# yvec \rangle \langle y \# yvec \rangle \langle x \# (p \cdot yvec) \rangle \langle y \# (p \cdot yvec) \rangle Sp Sq$
show *?thesis*
apply(*simp add: eqvts pt2[OF pt-name-inst]*)
by(*subst perm-compose[of q], simp*) +
qed
moreover from $\langle x \in supp N \rangle$ **have** $((q @ (x, y) \# p) \cdot x) \in ((q @ (x, y) \# p) \cdot (supp N))$
by(*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle y \# xvec \rangle \langle y \# (q \cdot xvec) \rangle Sp Sq$
have $y \in supp((q @ (x, y) \# p) \cdot N)$ **by**(*simp add: pt2[OF pt-name-inst] calc-atm eqvts*)
moreover from $\langle distinct xvec \rangle$ **have** $distinct(q \cdot xvec)$ **by** *simp*

```

moreover from  $\langle \text{distinct } yvec \rangle$  have  $\text{distinct}(p \cdot yvec)$  by simp
moreover note  $\langle x \# (q \cdot xvec) \rangle \langle x \# (p \cdot yvec) \rangle \langle x \# M \rangle \langle x \# \Psi \rangle$ 
 $\langle (q \cdot xvec) \#* \Psi \rangle \langle (q \cdot xvec) \#* P \rangle \langle (q \cdot xvec) \#* M \rangle \langle (q \cdot xvec) \#* (p \cdot yvec) \rangle$ 
 $\langle (p \cdot yvec) \#* \Psi \rangle \langle (p \cdot yvec) \#* P \rangle \langle (p \cdot yvec) \#* M \rangle \langle y \# (q \cdot xvec) \rangle \langle y \# (p \cdot$ 
 $yvec) \rangle \langle y \# M \rangle \langle y \# C \rangle \langle y \# \Psi \rangle$ 
 $\langle (p \cdot yvec) \#* C \rangle \langle (q \cdot xvec) \#* C \rangle$ 
ultimately show Prop by  $-(\text{rule } rBrOpen, (\text{assumption} \mid \text{simp})+)$ 
next
case(cScope  $P \alpha P' x$ )
obtain  $p::\text{name prm}$  where  $(bn(p \cdot \alpha)) \#* \Psi$  and  $(bn(p \cdot \alpha)) \#* P$ 
and  $(bn(p \cdot \alpha)) \#* \alpha$  and  $(bn(p \cdot \alpha)) \#* P'$  and  $x \# bn(p \cdot \alpha)$ 
and distinctPerm  $p$ 
and  $(bn(p \cdot \alpha)) \#* C$  and  $Sp: (\text{set } p) \subseteq \text{set}(bn \alpha) \times (\text{set}(bn(p \cdot \alpha)))$ 
by(rule name-list-avoiding[where  $xvec=bn \alpha$  and  $c=(\Psi, P, \alpha, x, P', C)$ ])
(auto simp add: eqvts)
obtain  $y::\text{name}$  where  $y \# \Psi$  and  $y \# P$  and  $y \# (p \cdot P')$  and  $y \# (p \cdot \alpha)$  and
 $y \# C$ 
by(generate-fresh name) (auto simp add: freshChainSimps simp del: action-Fresh)
from  $\langle bn \alpha \#* \text{subject } \alpha \rangle \langle \text{distinctPerm } p \rangle$  have  $bn(p \cdot \alpha) \#* \text{subject}(p \cdot \alpha)$ 
by(subst fresh-star-bij[symmetric, of - - p]) (simp add: eqvts)
from  $\langle \text{distinct}(bn \alpha) \rangle \langle \text{distinctPerm } p \rangle$  have  $\text{distinct}(bn(p \cdot \alpha))$ 
by(subst distinctClosed[symmetric, of - p]) (simp add: eqvts)
from  $\langle x \# \alpha \rangle \langle x \# (bn(p \cdot \alpha)) \rangle \langle \text{distinctPerm } p \rangle Sp$  have  $x \# (p \cdot \alpha)$ 
by(subst fresh-bij[symmetric, of - - p]) (simp add: eqvts freshChainSimps)

from  $\langle cP = (\nu x)P \rangle \langle y \# P \rangle$  have  $cP = (\nu y)([(x, y)] \cdot P)$  by(simp add: alphaRes)
moreover from  $\langle cRs = \alpha \prec (\nu x)P' \rangle \langle bn \alpha \#* \text{subject } \alpha \rangle \langle (bn(p \cdot \alpha)) \#* \alpha \rangle \langle x$ 
 $\# bn(p \cdot \alpha) \rangle \langle (bn(p \cdot \alpha)) \#* P' \rangle \langle x \# \alpha \rangle Sp$ 
have  $cRs = (p \cdot \alpha) \prec (\nu x)(p \cdot P')$ 
by(force simp add: residualAlpha)
with  $\langle y \# (p \cdot P') \rangle$  have  $cRs = (p \cdot \alpha) \prec (\nu y)([(x, y)] \cdot p \cdot P')$ 
by(simp add: alphaRes)
moreover from  $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle bn \alpha \#* \text{subject } \alpha \rangle \langle (bn(p \cdot \alpha)) \#* \alpha \rangle$ 
 $\langle (bn(p \cdot \alpha)) \#* P' \rangle Sp$ 
have  $\Psi \triangleright P \mapsto (p \cdot \alpha) \prec (p \cdot P')$  by(force simp add: residualAlpha)
then have( $[(x, y)] \cdot \Psi \triangleright [(x, y)] \cdot P \mapsto [(x, y)] \cdot ((p \cdot \alpha) \prec (p \cdot P'))$ )
by(rule eqvts)
with  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle y \# (p \cdot \alpha) \rangle \langle x \# (p \cdot \alpha) \rangle Sp \langle \text{distinctPerm } p \rangle$ 
have  $\Psi \triangleright ([x, y] \cdot P) \mapsto (p \cdot \alpha) \prec ([x, y] \cdot p \cdot P')$ 
by(simp add: eqvts)
moreover from  $\langle bn(p \cdot \alpha) \#* P \rangle \langle y \# (p \cdot \alpha) \rangle \langle y \# P \rangle$  have  $bn(p \cdot \alpha) \#* ([x,$ 
 $y]) \cdot P$ )
by(auto simp add: fresh-star-def fresh-left calc-atm) (simp add: fresh-def name-list-supp)
moreover from  $\langle \text{distinct}(bn \alpha) \rangle$  have  $\text{distinct}(p \cdot bn \alpha)$  by simp
then have  $\text{distinct}(bn(p \cdot \alpha))$  by(simp add: eqvts)
ultimately show ?thesis
using  $\langle y \# \Psi \rangle \langle y \# (p \cdot \alpha) \rangle \langle y \# C \rangle \langle bn(p \cdot \alpha) \#* \Psi \rangle \langle bn(p \cdot \alpha) \#* \text{subject}(p \cdot$ 
 $\alpha) \rangle \langle bn(p \cdot \alpha) \#* C \rangle$ 

```

by(*metis rScope*)
 next
 case(*Bang P*)
 then show *?thesis* by(*metis rBang*)
 qed

nominal-primrec

inputLength :: ('a::fs-name, 'b::fs-name, 'c::fs-name) *psi* \Rightarrow *nat*
 and *inputLength'* :: ('a::fs-name, 'b::fs-name, 'c::fs-name) *input* \Rightarrow *nat*
 and *inputLength''* :: ('a::fs-name, 'b::fs-name, 'c::fs-name) *psiCase* \Rightarrow *nat*

where

inputLength (**0**) = 0
 | *inputLength* (*M(N).P*) = 0
 | *inputLength* (*M(I)*) = *inputLength' I*
 | *inputLength* (*Case C*) = 0
 | *inputLength* (*P || Q*) = 0
 | *inputLength* ($(\nu x)P$) = 0
 | *inputLength* ($\{\Psi\}$) = 0
 | *inputLength* (*!P*) = 0

 | *inputLength'* (*Trm M P*) = 0
 | *inputLength'* ($\nu y I$) = 1 + (*inputLength' I*)

 | *inputLength''* (\perp_c) = 0
 | *inputLength''* ($\Box\Phi \Rightarrow P C$) = 0
 apply(*finite-guess*) +
 apply(*rule TrueI*) +
 by(*fresh-guess add: fresh-nat*) +

nominal-primrec *boundOutputLength* :: ('a, 'b, 'c) *boundOutput* \Rightarrow *nat*

where

boundOutputLength (*BOut M P*) = 0
 | *boundOutputLength* (*BStep x B*) = (*boundOutputLength B*) + 1
 apply(*finite-guess*) +
 apply(*rule TrueI*) +
 by(*fresh-guess add: fresh-nat*) +

nominal-primrec *residualLength* :: ('a, 'b, 'c) *residual* \Rightarrow *nat*

where

residualLength (*RIn M N P*) = 0
 | *residualLength* (*RBrIn M N P*) = 0
 | *residualLength* (*ROut M B*) = *boundOutputLength B*
 | *residualLength* (*RBrOut M B*) = *boundOutputLength B*
 | *residualLength* (*RTau P*) = 0
 by(*rule TrueI*) +

lemma *inputLengthProc[simp]*:

shows *inputLength*(*M*($\lambda *xvec N$).*P*) = *length xvec*

```

by(induct xvec) auto

lemma boundOutputLengthSimp[simp]:
  shows residualLength( $M(\nu * xvec) \langle N \rangle \prec P$ ) = length xvec
  and residualLength( $\downarrow M(\nu * xvec) \langle N \rangle \prec P$ ) = length xvec
by(induct xvec) (auto simp add: residualInject)

lemma boundOutputLengthSimp2[simp]:
  shows residualLength( $\alpha \prec P$ ) = length(bn  $\alpha$ )
by(nominal-induct  $\alpha$  rule: action.strong-induct, auto) (auto simp add: residual-Inject)

lemmas [simp del] = inputLength-inputLength'-inputLength''.simps residualLength.simps
boundOutputLength.simps

lemma constructPerm:
  fixes xvec :: name list
  and yvec :: name list

assumes length xvec = length yvec
  and xvec #* yvec
  and distinct xvec
  and distinct yvec

obtains p where set p  $\subseteq$  set xvec  $\times$  set(p  $\cdot$  xvec) and distinctPerm p and yvec
= p  $\cdot$  xvec
proof -
  assume  $\bigwedge p. \llbracket \text{set } p \subseteq \text{set } xvec \times \text{set } (p \cdot xvec); \text{distinctPerm } p; yvec = p \cdot xvec \rrbracket$ 
 $\implies$  thesis
  moreover obtain n where n = length xvec by auto
  with assms have  $\exists p. (\text{set } p) \subseteq (\text{set } xvec) \times \text{set } (yvec) \wedge \text{distinctPerm } p \wedge yvec$ 
= p  $\cdot$  xvec
  proof(induct n arbitrary: xvec yvec)
    case(0 xvec yvec)
    then show ?case by simp
  next
    case(Suc n xvec yvec)
    from  $\langle \text{Suc } n = \text{length } xvec \rangle$ 
    obtain x xvec' where xvec = x#xvec' and length xvec' = n
    by(cases xvec) auto
    from  $\langle \text{length } xvec = \text{length } yvec \rangle \langle xvec = x \# xvec' \rangle$ 
    obtain y yvec' where length xvec' = length yvec' and yvec = y#yvec'
    by(cases yvec) auto
    from  $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle \langle xvec \#* yvec \rangle$ 
    have  $x \neq y$  and xvec' #* yvec' and x # yvec' and y # xvec'
    by(auto simp add: fresh-list-cons)
    from  $\langle \text{distinct } xvec \rangle \langle \text{distinct } yvec \rangle \langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$  have x #
xvec' and y # yvec' and distinct xvec' and distinct yvec'
    by simp+

```

from $\langle \text{Suc } n = \text{length } xvec \rangle \langle xvec = x \# xvec' \rangle$ **have** $n = \text{length } xvec'$ **by** *simp*
with $\langle \text{length } xvec' = \text{length } yvec' \rangle \langle xvec' \#* yvec' \rangle \langle \text{distinct } xvec' \rangle \langle \text{distinct } yvec' \rangle$
obtain p **where** $S: \text{set } p \subseteq \text{set } xvec' \times \text{set } yvec'$ **and** *distinctPerm* p **and** $yvec' = p \cdot xvec'$
by $-$ (*drule* *Suc, auto*)
from S **have** $\text{set}((x, y) \# p) \subseteq \text{set}(x \# xvec') \times \text{set}(y \# yvec')$ **by** *auto*
moreover from $\langle x \# xvec' \rangle \langle x \# yvec' \rangle \langle y \# xvec' \rangle \langle y \# yvec' \rangle$ S **have** $x \# p$ **and**
 $y \# p$
apply (*induct* p)
by (*clarsimp simp add: fresh-list-nil fresh-list-cons fresh-prod name-list-supp; force simp add: fresh-def*) $+$

with $S \langle \text{distinctPerm } p \rangle \langle x \neq y \rangle$ **have** *distinctPerm* $((x, y) \# p)$ **by** *auto*
moreover from $\langle yvec' = p \cdot xvec' \rangle \langle x \# p \rangle \langle y \# p \rangle \langle x \# xvec' \rangle \langle y \# yvec' \rangle$ **have**
 $(y \# yvec') = ((x, y) \# p) \cdot (x \# xvec')$
by (*simp add: calc-atm freshChainSimps*)
ultimately show $?case$ **using** $\langle xvec = x \# xvec' \rangle \langle yvec = y \# yvec' \rangle$
by *blast*
qed
ultimately show $?thesis$ **by** *blast*
qed

lemma *distinctApend[simp]*:
fixes $xvec :: \text{name list}$
and $yvec :: \text{name list}$

shows $(\text{set } xvec \cap \text{set } yvec = \{\}) = xvec \#* yvec$
by (*auto simp add: fresh-star-def name-list-supp fresh-def*)

lemma *lengthAux*:
fixes $xvec :: \text{name list}$
and $y :: \text{name}$
and $yvec :: \text{name list}$

assumes $\text{length } xvec = \text{length}(y \# yvec)$

obtains z $zvec$ **where** $xvec = z \# zvec$ **and** $\text{length } zvec = \text{length } yvec$
using *assms*
by (*induct xvec arbitrary: yvec y*) *auto*

lemma *lengthAux2*:
fixes $xvec :: \text{name list}$
and $yvec :: \text{name list}$
and $zvec :: \text{name list}$

assumes $\text{length } xvec = \text{length}(yvec @ y \# zvec)$

obtains $xvec1$ $xvec2$ **where** $xvec = xvec1 @ x \# xvec2$ **and** $\text{length } xvec1 = \text{length } yvec$
and $\text{length } xvec2 = \text{length } zvec$

proof –
assume $\bigwedge xvec1\ x\ xvec2.$
 $\llbracket xvec = xvec1\ @\ x\ \#\ xvec2; \text{length } xvec1 = \text{length } yvec;$
 $\text{length } xvec2 = \text{length } zvec \rrbracket$
 $\implies \text{thesis}$
moreover from *assms* **have** $\exists xvec1\ x\ xvec2. xvec = xvec1 @ x \# xvec2 \wedge \text{length } xvec1 = \text{length } yvec \wedge \text{length } xvec2 = \text{length } zvec$
apply –
apply(*rule exI*[**where** $x = \text{take } (\text{length } yvec)\ xvec$])
apply(*rule exI*[**where** $x = \text{hd}(\text{drop } (\text{length } yvec)\ xvec)$])
apply(*rule exI*[**where** $x = \text{tl}(\text{drop } (\text{length } yvec)\ xvec)$])
by *auto*
ultimately show *?thesis* **by** *blast*
qed

lemma *semanticsCases*[*consumes 19, case-names cInput cBrInput cOutput cBrOutput cCase cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBrClose cOpen cBrOpen cScope cBang*]:

fixes $\Psi :: 'b$
and $cP :: ('a, 'b, 'c)\ \text{psi}$
and $cRs :: ('a, 'b, 'c)\ \text{residual}$
and $C :: 'f::\text{fs-name}$
and $x1 :: \text{name}$
and $x2 :: \text{name}$
and $x3 :: \text{name}$
and $x4 :: \text{name}$
and $xvec1 :: \text{name list}$
and $xvec2 :: \text{name list}$
and $xvec3 :: \text{name list}$
and $xvec4 :: \text{name list}$
and $xvec5 :: \text{name list}$
and $xvec6 :: \text{name list}$
and $xvec7 :: \text{name list}$
and $xvec8 :: \text{name list}$
and $xvec9 :: \text{name list}$

assumes $\Psi \triangleright cP \mapsto cRs$

and $\text{length } xvec1 = \text{inputLength } cP$ **and** *distinct* $xvec1$
and $\text{length } xvec6 = \text{inputLength } cP$ **and** *distinct* $xvec6$
and $\text{length } xvec2 = \text{residualLength } cRs$ **and** *distinct* $xvec2$
and $\text{length } xvec3 = \text{residualLength } cRs$ **and** *distinct* $xvec3$
and $\text{length } xvec4 = \text{residualLength } cRs$ **and** *distinct* $xvec4$
and $\text{length } xvec5 = \text{residualLength } cRs$ **and** *distinct* $xvec5$
and $\text{length } xvec7 = \text{residualLength } cRs$ **and** *distinct* $xvec7$
and $\text{length } xvec8 = \text{residualLength } cRs$ **and** *distinct* $xvec8$
and $\text{length } xvec9 = \text{residualLength } cRs$ **and** *distinct* $xvec9$
and $rInput: \bigwedge M\ K\ N\ Tvec\ P. (\llbracket xvec1\ \#\ \Psi; xvec1\ \#\ cP; xvec1\ \#\ cRs \rrbracket \implies cP = M(\lambda * xvec1\ N).P \wedge cRs = K(\llbracket N[xvec1 ::= Tvec] \rrbracket) \prec P[xvec1 ::= Tvec] \wedge \Psi \vdash M \leftrightarrow K \wedge \text{distinct } xvec1 \wedge \text{set } xvec1 \subseteq$

$supp\ N \wedge length\ xvec1=length\ Tvec \wedge$
 $xvec1 \#* Tvec \wedge xvec1 \#* \Psi \wedge xvec1 \#* M \wedge$
 $xvec1 \#* K) \implies Prop$
and $rBrInput: \bigwedge M\ K\ N\ Tvec\ P. (\llbracket xvec6 \#* \Psi; xvec6 \#* cP; xvec6 \#* cRs \rrbracket \implies$
 $cP = M(\lambda*xvec6\ N).P \wedge cRs = \downarrow K(\langle N[xvec6::=Tvec] \rangle) \prec P[xvec6::=Tvec] \wedge$
 $\Psi \vdash K \succeq M \wedge distinct\ xvec6 \wedge set\ xvec6 \subseteq$
 $supp\ N \wedge length\ xvec6=length\ Tvec \wedge$
 $xvec6 \#* Tvec \wedge xvec6 \#* \Psi \wedge xvec6 \#* M \wedge$
 $xvec6 \#* K) \implies Prop$
and $rOutput: \bigwedge M\ K\ N\ P. \llbracket cP = M\langle N \rangle.P; cRs = K\langle N \rangle \prec P; \Psi \vdash M \leftrightarrow K \rrbracket$
 $\implies Prop$
and $rBrOutput: \bigwedge M\ K\ N\ P. \llbracket cP = M\langle N \rangle.P; cRs = \downarrow K\langle N \rangle \prec P; \Psi \vdash M \preceq$
 $K \rrbracket \implies Prop$
and $rCase: \bigwedge Cs\ P\ \varphi. \llbracket cP = Cases\ Cs; \Psi \triangleright P \mapsto cRs; (\varphi, P) \in set\ Cs; \Psi \vdash$
 $\varphi; guarded\ P \rrbracket \implies Prop$
and $rPar1: \bigwedge \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q. (\llbracket xvec2 \#* \Psi; xvec2 \#* cP; xvec2 \#* cRs \rrbracket \implies$
 $cP = P \parallel Q \wedge cRs = \alpha \prec (P' \parallel Q) \wedge xvec2 =$
 $bn\ \alpha \wedge$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \wedge extractFrame\ Q =$
 $\langle A_Q, \Psi_Q \rangle \wedge distinct\ A_Q \wedge$
 $A_Q \#* P \wedge A_Q \#* Q \wedge A_Q \#* \Psi \wedge A_Q \#* \alpha \wedge$
 $A_Q \#* P' \wedge A_Q \#* C) \implies Prop$
and $rPar2: \bigwedge \Psi_P\ Q\ \alpha\ Q'\ P\ A_P. (\llbracket xvec3 \#* \Psi; xvec3 \#* cP; xvec3 \#* cRs \rrbracket \implies$
 $cP = P \parallel Q \wedge cRs = \alpha \prec (P \parallel Q') \wedge xvec3 =$
 $bn\ \alpha \wedge$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \wedge extractFrame\ P =$
 $\langle A_P, \Psi_P \rangle \wedge distinct\ A_P \wedge$
 $A_P \#* P \wedge A_P \#* Q \wedge A_P \#* \Psi \wedge A_P \#* \alpha \wedge$
 $A_P \#* Q' \wedge A_P \#* C) \implies Prop$
and $rComm1: \bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q.$
 $\llbracket cP = P \parallel Q; cRs = \tau \prec (\nu*xvec)P' \parallel Q';$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle N \rangle) \prec P'; extractFrame\ P = \langle A_P, \Psi_P \rangle;$
 $distinct\ A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu*xvec)\langle N \rangle \prec Q'; extractFrame\ Q = \langle A_Q,$
 $\Psi_Q \rangle; distinct\ A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#*$
 $M; A_P \#* N;$
 $A_P \#* P'; A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi;$
 $A_Q \#* \Psi_P;$
 $A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; A_Q$
 $\#* xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#*$
 $Q;$
 $xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C; distinct\ xvec \rrbracket \implies Prop$
and $rComm2: \bigwedge \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q.$
 $\llbracket cP = P \parallel Q; cRs = \tau \prec (\nu*xvec)P' \parallel Q';$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; extractFrame\ P = \langle A_P,$
 $\Psi_P \rangle; distinct\ A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\langle N \rangle) \prec Q'; extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$

$distinct\ A_Q;$
 $M; A_P \#* N;$
 $A_Q \#* \Psi_P;$
 $\#* xvec;$
 $Q;$

$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#*$
 $A_P \#* P'; A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* xvec; A_Q \#* \Psi;$
 $A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* Q'; A_Q$
 $\#* xvec;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* \Psi_Q; xvec \#* P; xvec \#* M; xvec \#*$
 $xvec \#* K; A_P \#* C; A_Q \#* C; xvec \#* C; distinct\ xvec \implies Prop$

and $rBrMerge: \bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q.$
 $\llbracket cP = (P \parallel Q); cRs = \imath M(\downarrow N) \prec (P' \parallel Q') \rrbracket;$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\downarrow N) \prec P'; extractFrame\ P = \langle A_P, \Psi_P \rangle;$

$distinct\ A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\downarrow N) \prec Q'; extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;$

$distinct\ A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C \implies Prop$

and $rBrComm1: \bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q.$
 $\llbracket xvec7 \#* \Psi; xvec7 \#* cP; xvec7 \#* cRs \rrbracket \implies$
 $cP = P \parallel Q \wedge cRs = \imath M(\nu * xvec7)(\downarrow N) \prec (P' \parallel Q') \wedge$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\downarrow N) \prec P' \wedge extractFrame\ P = \langle A_P, \Psi_P \rangle \wedge$

$distinct\ A_P \wedge$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\nu * xvec7)(\downarrow N) \prec Q' \wedge extractFrame\ Q =$
 $\langle A_Q, \Psi_Q \rangle \wedge distinct\ A_Q \wedge$
 $A_P \#* \Psi \wedge A_P \#* \Psi_Q \wedge A_P \#* P \wedge A_P \#* N \wedge$
 $A_P \#* P' \wedge A_P \#* Q \wedge A_P \#* Q' \wedge A_P \#* A_Q \wedge A_P \#* xvec7 \wedge$
 $A_Q \#* \Psi \wedge A_Q \#* \Psi_P \wedge$
 $A_Q \#* P \wedge A_Q \#* N \wedge A_Q \#* P' \wedge A_Q \#* Q \wedge A_Q \#* Q' \wedge A_Q \#*$
 $xvec7 \wedge$
 $xvec7 \#* \Psi \wedge xvec7 \#* \Psi_P \wedge xvec7 \#* \Psi_Q \wedge xvec7 \#* P \wedge xvec7$
 $\#* Q \wedge$
 $A_P \#* M \wedge A_Q \#* M \wedge xvec7 \#* M \wedge$
 $A_P \#* C \wedge A_Q \#* C \wedge distinct\ xvec7 \implies Prop$

and $rBrComm2: \bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ Q'\ A_Q.$
 $\llbracket xvec8 \#* \Psi; xvec8 \#* cP; xvec8 \#* cRs \rrbracket \implies$
 $cP = P \parallel Q \wedge cRs = \imath M(\nu * xvec8)(\downarrow N) \prec (P' \parallel Q') \wedge$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\nu * xvec8)(\downarrow N) \prec P' \wedge extractFrame\ P = \langle A_P,$
 $\Psi_P \rangle \wedge distinct\ A_P \wedge$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\downarrow N) \prec Q' \wedge extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \wedge$
 $distinct\ A_Q \wedge$
 $A_P \#* \Psi \wedge A_P \#* \Psi_Q \wedge A_P \#* P \wedge A_P \#* N \wedge$
 $A_P \#* P' \wedge A_P \#* Q \wedge A_P \#* Q' \wedge A_P \#* A_Q \wedge A_P \#* xvec8 \wedge$
 $A_Q \#* \Psi \wedge A_Q \#* \Psi_P \wedge$
 $A_Q \#* P \wedge A_Q \#* N \wedge A_Q \#* P' \wedge A_Q \#* Q \wedge A_Q \#* Q' \wedge A_Q \#*$
 $xvec8 \wedge$
 $xvec8 \#* \Psi \wedge xvec8 \#* \Psi_P \wedge xvec8 \#* \Psi_Q \wedge xvec8 \#* P \wedge xvec8$

$\#* Q \wedge$
 $A_P \#* M \wedge A_Q \#* M \wedge xvec8 \#* M \wedge$
 $A_P \#* C \wedge A_Q \#* C \wedge distinct\ xvec8) \implies Prop$
and $rBrClose: \bigwedge P M N\ xvec\ P'.$
 $(\llbracket x3 \# \Psi; x3 \# cP; x3 \# cRs \rrbracket \implies$
 $cP = (\nu x3)P \wedge cRs = \tau \prec (\nu x3)((\nu *xvec)P') \wedge$
 $x3 \in supp\ M \wedge$
 $\Psi \triangleright P \mapsto iM(\nu *xvec)\langle N \rangle \prec P' \wedge$
 $distinct\ xvec \wedge xvec \#* \Psi \wedge xvec \#* P \wedge$
 $xvec \#* M \wedge xvec \#* C \wedge$
 $x3 \# \Psi \wedge x3 \# xvec) \implies Prop$
and $rOpen: \bigwedge P M\ xvec\ y\ yvec\ N\ P'.$
 $(\llbracket xvec4 \#* \Psi; xvec4 \#* cP; xvec4 \#* cRs; x1 \# \Psi; x1 \# cP; x1 \#$
 $cRs; x1 \# xvec4 \rrbracket \implies$
 $cP = (\nu x1)P \wedge cRs = M(\nu *(xvec@x1\#yvec))\langle N \rangle \prec P' \wedge$
 $xvec4 = xvec@y\#yvec \wedge$
 $\Psi \triangleright P \mapsto M(\nu *(xvec@yvec))\langle N \rangle \prec P' \wedge x1 \in supp\ N \wedge x1 \#$
 $xvec \wedge x1 \# yvec \wedge$
 $distinct\ xvec \wedge distinct\ yvec \wedge xvec \#* \Psi \wedge xvec \#* P \wedge xvec \#* M$
 $\wedge xvec \#* yvec \wedge$
 $yvec \#* \Psi \wedge yvec \#* P \wedge yvec \#* M) \implies Prop$
and $rBrOpen: \bigwedge P M\ xvec\ y\ yvec\ N\ P'.$
 $(\llbracket xvec9 \#* \Psi; xvec9 \#* cP; xvec9 \#* cRs; x4 \# \Psi; x4 \# cP; x4 \#$
 $cRs; x4 \# xvec9 \rrbracket \implies$
 $cP = (\nu x4)P \wedge cRs = iM(\nu *(xvec@x4\#yvec))\langle N \rangle \prec P' \wedge$
 $xvec9 = xvec@y\#yvec \wedge$
 $\Psi \triangleright P \mapsto iM(\nu *(xvec@yvec))\langle N \rangle \prec P' \wedge x4 \in supp\ N \wedge x4 \#$
 $xvec \wedge x4 \# yvec \wedge$
 $distinct\ xvec \wedge distinct\ yvec \wedge xvec \#* \Psi \wedge xvec \#* P \wedge xvec \#* M$
 $\wedge xvec \#* yvec \wedge$
 $yvec \#* \Psi \wedge yvec \#* P \wedge yvec \#* M) \implies Prop$
and $rScope: \bigwedge P\ \alpha\ P'. (\llbracket xvec5 \#* \Psi; xvec5 \#* cP; xvec5 \#* cRs; x2 \# \Psi; x2 \#$
 $cP; x2 \# cRs; x2 \# xvec5 \rrbracket \implies$
 $cP = (\nu x2)P \wedge cRs = \alpha \prec (\nu x2)P' \wedge xvec5 = bn\ \alpha \wedge$
 $\Psi \triangleright P \mapsto \alpha \prec P' \wedge x2 \# \Psi \wedge x2 \# \alpha \wedge bn\ \alpha \#* subject$
 $\alpha \wedge distinct(bn\ \alpha)) \implies Prop$
and $rBang: \bigwedge P. \llbracket cP = !P;$
 $\Psi \triangleright P \parallel !P \mapsto cRs; guarded\ P \rrbracket \implies Prop$

shows $Prop$
using $\langle \Psi \triangleright cP \mapsto cRs \rangle$
proof(*cases rule: semanticsCasesAux*[**where** $C = (xvec1, xvec2, xvec3, xvec4, xvec5,$
 $xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C)$])
case(*cInput* $M\ K\ xvec\ N\ Tvec\ P$)
have $B: cP = M(\lambda *xvec\ N).P$ **and** $C: cRs = K(\langle N[xvec ::= Tvec] \rangle) \prec (P[xvec ::= Tvec])$
by *fact+*
from $\langle xvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,$
 $x2, x3, x4, cP, cRs, C) \rangle$ **have** $xvec \#* xvec1$ **by** *simp*

from $\langle length\ xvec1 = inputLength\ cP \rangle$ B **have** $length\ xvec1 = length\ xvec$

by simp
then obtain p where $S: \text{set } p \subseteq \text{set } \text{vec} \times \text{set}(p \cdot \text{vec})$ and $\text{distinctPerm } p$
and $\text{vec1} = p \cdot \text{vec}$
using $\langle \text{vec} \#* \text{vec1} \rangle \langle \text{distinct } \text{vec} \rangle \langle \text{distinct } \text{vec1} \rangle$
by $-$ (rule $\text{constructPerm}[\text{where } \text{vec}=\text{vec} \text{ and } \text{yvec}=\text{vec1}], \text{ auto}$)
show ?thesis
proof(rule $r\text{Input}[\text{where } M=M \text{ and } K=K \text{ and } N = p \cdot N \text{ and } T\text{vec}=T\text{vec}$
and $P=p \cdot P]$)
assume $\text{vec1} \#* \Psi$ and $\text{vec1} \#* cP$ and $\text{vec1} \#* cRs$
from $B \langle \text{vec} \#* \text{vec1} \rangle \langle \text{vec1} \#* cP \rangle$ have $\text{vec1} \#* N$ and $\text{vec1} \#* P$
by(auto simp add: fresh-star-def inputChainFresh name-list-supp) (auto simp
add: fresh-def)
moreover from $\langle cP = M(\lambda*\text{vec } N).P \rangle S \langle \text{vec1} \#* N \rangle \langle \text{vec1} \#* P \rangle \langle \text{vec1}$
 $= p \cdot \text{vec} \rangle$
have $cP = M(\lambda*\text{vec1 } (p \cdot N)).(p \cdot P)$
apply simp
by(subst inputChainAlpha) auto
moreover from $\langle cRs = K((N[\text{vec}::=T\text{vec}])) \rangle \prec P[\text{vec}::=T\text{vec}] \rangle S \langle \text{vec1} \#*$
 $N \rangle \langle \text{vec1} \#* P \rangle \langle \text{vec1} = p \cdot \text{vec} \rangle \langle \text{length } \text{vec} = \text{length } T\text{vec} \rangle \langle \text{distinctPerm } p \rangle$
have $cRs = K((p \cdot N)[\text{vec1}::=T\text{vec}])) \rangle \prec (p \cdot P)[\text{vec1}::=T\text{vec}]$
by(simp add: renaming substTerm.renaming)
moreover note $\langle \Psi \vdash M \leftrightarrow K \rangle$
moreover from $\langle \text{distinct } \text{vec} \rangle \langle \text{vec1} = p \cdot \text{vec} \rangle$ have $\text{distinct } \text{vec1}$ by simp
moreover from $\langle \text{set } \text{vec} \subseteq \text{supp } N \rangle$ have $(p \cdot \text{set } \text{vec}) \subseteq (p \cdot (\text{supp } N))$
by(simp add: eqvts)
with $\langle \text{vec1} = p \cdot \text{vec} \rangle$ have $\text{set } \text{vec1} \subseteq \text{supp}(p \cdot N)$ by(simp add: eqvts)
moreover from $\langle \text{length } \text{vec} = \text{length } T\text{vec} \rangle \langle \text{vec1} = p \cdot \text{vec} \rangle$ have length
 $\text{vec1} = \text{length } T\text{vec}$
by simp

moreover from $\langle \text{vec1} \#* cRs \rangle C \langle \text{length } \text{vec} = \text{length } T\text{vec} \rangle \langle \text{distinct } \text{vec} \rangle$
 $\langle \text{set } \text{vec} \subseteq \text{supp } N \rangle$
have $(\text{set } \text{vec1}) \#* T\text{vec}$
by $-$ (rule $\text{substTerm.subst3Chain}[\text{where } T=N], \text{ auto}$)
then have $\text{vec1} \#* T\text{vec}$ by simp
moreover from $\langle \text{vec} \#* T\text{vec} \rangle$ have $(p \cdot \text{vec}) \#* (p \cdot T\text{vec})$ by(simp add:
fresh-star-bij)
with $S \langle \text{vec} \#* T\text{vec} \rangle \langle \text{vec1} \#* T\text{vec} \rangle \langle \text{vec1} = p \cdot \text{vec} \rangle$ have $\text{vec1} \#* T\text{vec}$
by simp
moreover note $\langle \text{vec1} \#* \Psi \rangle$
moreover from $\langle \text{vec} \#* M \rangle$ have $(p \cdot \text{vec}) \#* (p \cdot M)$ by(simp add:
fresh-star-bij)
with $S \langle \text{vec} \#* M \rangle \langle \text{vec1} \#* cP \rangle B \langle \text{vec1} = p \cdot \text{vec} \rangle$ have $\text{vec1} \#* M$ by
simp
moreover from $\langle \text{vec} \#* K \rangle$ have $(p \cdot \text{vec}) \#* (p \cdot K)$ by(simp add:
fresh-star-bij)
with $S \langle \text{vec} \#* K \rangle \langle \text{vec1} \#* cRs \rangle C \langle \text{vec1} = p \cdot \text{vec} \rangle$ have $\text{vec1} \#* K$ by
simp
ultimately show $cP = M(\lambda*\text{vec1 } p \cdot N).p \cdot P \wedge cRs = K((p \cdot N)[\text{vec1}::=T\text{vec}])$

$\prec (p \cdot P)[xvec1 ::= Tvec] \wedge$
 $\Psi \vdash M \leftrightarrow K \wedge distinct\ xvec1 \wedge set\ xvec1 \subseteq supp\ (p \cdot N) \wedge length\ xvec1 =$
 $length\ Tvec \wedge$
 $xvec1 \#* Tvec \wedge xvec1 \#* \Psi \wedge xvec1 \#* M \wedge xvec1 \#* K$
by *blast*
qed
next
case(*cBrInput* $M\ K\ xvec\ N\ Tvec\ P$)
have $B: cP = M(\lambda * xvec\ N).P$ **and** $C: cRs = \iota K((N[xvec ::= Tvec])) \prec (P[xvec ::= Tvec])$
by *fact+*
from $\langle xvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,$
 $x2, x3, x4, cP, cRs, C) \rangle$ **have** $xvec \#* xvec6$ **by** *simp*

from $\langle length\ xvec6 = inputLength\ cP \rangle B$ **have** $length\ xvec6 = length\ xvec$
by *simp*
then obtain p **where** $S: set\ p \subseteq set\ xvec \times set(p \cdot xvec)$ **and** *distinctPerm* p
and $xvec6 = p \cdot xvec$
using $\langle xvec \#* xvec6 \rangle \langle distinct\ xvec \rangle \langle distinct\ xvec6 \rangle$
by $- (rule\ constructPerm[where\ xvec=xvec\ and\ yvec=xvec6], auto)$
show *?thesis*
proof(*rule* *rBrInput*[**where** $M=M$ **and** $K=K$ **and** $N = p \cdot N$ **and** $Tvec=Tvec$
and $P=p \cdot P$])
assume $xvec6 \#* \Psi$ **and** $xvec6 \#* cP$ **and** $xvec6 \#* cRs$
from $B \langle xvec \#* xvec6 \rangle \langle xvec6 \#* cP \rangle$ **have** $xvec6 \#* N$ **and** $xvec6 \#* P$
by(*auto* *simp* *add: fresh-star-def* *inputChainFresh* *name-list-supp*) (*auto* *simp*
add: fresh-def)

moreover from $\langle cP = M(\lambda * xvec\ N).P \rangle S \langle xvec6 \#* N \rangle \langle xvec6 \#* P \rangle \langle xvec6$
 $= p \cdot xvec \rangle$
have $cP = M(\lambda * xvec6\ (p \cdot N)).(p \cdot P)$
apply *simp*
by(*subst* *inputChainAlpha*) *auto*
moreover from $\langle cRs = \iota K((N[xvec ::= Tvec])) \prec P[xvec ::= Tvec] \rangle S \langle xvec6 \#*$
 $N \rangle \langle xvec6 \#* P \rangle \langle xvec6 = p \cdot xvec \rangle \langle length\ xvec = length\ Tvec \rangle \langle distinctPerm\ p \rangle$
have $cRs = \iota K(((p \cdot N)[xvec6 ::= Tvec])) \prec (p \cdot P)[xvec6 ::= Tvec]$
by(*simp* *add: renaming* *substTerm.renaming*)
moreover note $\langle \Psi \vdash K \succeq M \rangle$
moreover from $\langle distinct\ xvec \rangle \langle xvec6 = p \cdot xvec \rangle$ **have** *distinct* $xvec6$ **by** *simp*
moreover from $\langle set\ xvec \subseteq supp\ N \rangle$ **have** $(p \cdot set\ xvec) \subseteq (p \cdot (supp\ N))$
by(*simp* *add: eqts*)
with $\langle xvec6 = p \cdot xvec \rangle$ **have** $set\ xvec6 \subseteq supp(p \cdot N)$ **by**(*simp* *add: eqts*)
moreover from $\langle length\ xvec = length\ Tvec \rangle \langle xvec6 = p \cdot xvec \rangle$ **have** $length$
 $xvec6 = length\ Tvec$
by *simp*

moreover from $\langle xvec6 \#* cRs \rangle C \langle length\ xvec = length\ Tvec \rangle \langle distinct\ xvec \rangle$
 $\langle set\ xvec \subseteq supp\ N \rangle$
have $(set\ xvec6) \#* Tvec$
by $- (rule\ substTerm.subst3Chain[where\ T=N], auto)$

```

then have  $xvec6 \#* Tvec$  by simp
moreover from  $\langle xvec \#* Tvec \rangle$  have  $(p \cdot xvec) \#* (p \cdot Tvec)$  by(simp add:
fresh-star-bij)
with  $S \langle xvec \#* Tvec \rangle \langle xvec6 \#* Tvec \rangle \langle xvec6 = p \cdot xvec \rangle$  have  $xvec6 \#* Tvec$ 
by simp
moreover note  $\langle xvec6 \#* \Psi \rangle$ 
moreover from  $\langle xvec \#* M \rangle$  have  $(p \cdot xvec) \#* (p \cdot M)$  by(simp add:
fresh-star-bij)
with  $S \langle xvec \#* M \rangle \langle xvec6 \#* cP \rangle B \langle xvec6 = p \cdot xvec \rangle$  have  $xvec6 \#* M$  by
simp
moreover from  $\langle xvec \#* K \rangle$  have  $(p \cdot xvec) \#* (p \cdot K)$  by(simp add:
fresh-star-bij)
with  $S \langle xvec \#* K \rangle \langle xvec6 \#* cRs \rangle C \langle xvec6 = p \cdot xvec \rangle$  have  $xvec6 \#* K$  by
simp
ultimately show  $cP = M(\lambda xvec6 p \cdot N).p \cdot P \wedge$ 
 $cRs = \iota K((p \cdot N)[xvec6::=Tvec]) \prec (p \cdot P)[xvec6::=Tvec] \wedge$ 
 $\Psi \vdash K \succeq M \wedge distinct\ xvec6 \wedge set\ xvec6 \subseteq supp\ (p \cdot N) \wedge length\ xvec6 =$ 
 $length\ Tvec \wedge$ 
 $xvec6 \#* Tvec \wedge xvec6 \#* \Psi \wedge xvec6 \#* M \wedge xvec6 \#* K$  by blast
qed
next
case(cOutput M K N P)
then show ?thesis by(rule rOutput)
next
case(cBrOutput M N P)
then show ?thesis by(rule rBrOutput)
next
case(cCase Cs P  $\varphi$ )
then show ?thesis by(rule rCase)
next
case(cPar1  $\Psi_Q P \alpha P' Q A_Q$ )
have  $B: cP = P \parallel Q$  and  $C: cRs = \alpha \prec P' \parallel Q$ 
by fact+
from  $\langle bn\ \alpha \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9,$ 
 $x1, x2, x3, x4, cP, cRs, C) \rangle$  have  $bn\ \alpha \#* xvec2$  by simp
from  $\langle A_Q \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,$ 
 $x2, x3, x4, cP, cRs, C) \rangle$  have  $A_Q \#* xvec2$  and  $A_Q \#* C$  by simp+

from  $\langle length\ xvec2 = residualLength\ cRs \rangle C$  have  $length\ xvec2 = length(bn\ \alpha)$ 
by simp
then obtain  $p$  where  $S: set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$  and distinctPerm
 $p$  and  $xvec2 = bn(p \cdot \alpha)$ 
using  $\langle bn\ \alpha \#* xvec2 \rangle \langle distinct(bn\ \alpha) \rangle \langle distinct\ xvec2 \rangle$ 
by  $- (rule\ constructPerm[where\ xvec=bn\ \alpha\ and\ yvec=xvec2], auto\ simp\ add:$ 
eqts)
show ?thesis
proof(rule rPar1[where P=P and Q=Q and  $\alpha=p \cdot \alpha$  and  $P'=p \cdot P'$  and
 $A_Q=A_Q$  and  $\Psi_Q=\Psi_Q]$ )
assume  $xvec2 \#* \Psi$  and  $xvec2 \#* cP$  and  $xvec2 \#* cRs$ 

```

note $\langle cP = P \parallel Q \rangle$
moreover from $C\ S\ \langle bn\ \alpha\ \#* \ xvec2 \rangle\ \langle xvec2\ \#* \ cRs \rangle\ \langle xvec2 = bn(p \cdot \alpha) \rangle\ \langle bn\ \alpha\ \#* \ subject\ \alpha \rangle\ \langle xvec2\ \#* \ cP \rangle\ \langle bn\ \alpha\ \#* \ Q \rangle$
have $cRs = (p \cdot \alpha) \prec (p \cdot P') \parallel Q$
apply *clarsimp*
by(*subst residualAlpha*[**where** $p=p$]) *auto*
moreover note $\langle xvec2 = bn(p \cdot \alpha) \rangle$
moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle\ S\ B\ C\ S\ \langle bn\ \alpha\ \#* \ xvec2 \rangle\ \langle xvec2\ \#* \ cRs \rangle\ \langle xvec2 = bn(p \cdot \alpha) \rangle\ \langle bn\ \alpha\ \#* \ subject\ \alpha \rangle\ \langle xvec2\ \#* \ cP \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto (p \cdot \alpha) \prec (p \cdot P')$
by(*subst residualAlpha*[*symmetric*]) *auto*
moreover note $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle\ \langle distinct\ A_Q \rangle\ \langle A_Q\ \#* \ P \rangle\ \langle A_Q\ \#* \ Q \rangle\ \langle A_Q\ \#* \ \Psi \rangle\ \langle A_Q\ \#* \ \alpha \rangle$
moreover from $\langle A_Q\ \#* \ \alpha \rangle\ \langle A_Q\ \#* \ xvec2 \rangle\ S\ \langle xvec2 = bn(p \cdot \alpha) \rangle\ \langle distinctPerm\ p \rangle$
have $A_Q\ \#* \ (p \cdot \alpha)$
by(*subst fresh-star-bij*[*symmetric, where pi=p*]) *simp*
moreover from $\langle A_Q\ \#* \ P' \rangle\ \langle A_Q\ \#* \ \alpha \rangle\ \langle A_Q\ \#* \ xvec2 \rangle\ S\ \langle xvec2 = bn(p \cdot \alpha) \rangle\ \langle distinctPerm\ p \rangle$
have $A_Q\ \#* \ (p \cdot P')$
by(*subst fresh-star-bij*[*symmetric, where pi=p*]) *simp*
moreover note $\langle A_Q\ \#* \ C \rangle$
ultimately show $cP = P \parallel Q \wedge cRs = (p \cdot \alpha) \prec (p \cdot P') \parallel Q \wedge xvec2 = bn(p \cdot \alpha) \wedge \Psi \otimes \Psi_Q \triangleright P \mapsto (p \cdot \alpha) \prec p \cdot P' \wedge extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \wedge distinct\ A_Q \wedge A_Q\ \#* \ P \wedge A_Q\ \#* \ Q \wedge A_Q\ \#* \ \Psi \wedge A_Q\ \#* \ (p \cdot \alpha) \wedge A_Q\ \#* \ (p \cdot P') \wedge A_Q\ \#* \ C$
by blast qed
next
case(*cPar2* $\Psi_P\ Q\ \alpha\ Q'\ P\ A_P$)
have $B: cP = P \parallel Q$ **and** $C: cRs = \alpha \prec P \parallel Q'$
by fact+
from $\langle bn\ \alpha\ \#* \ (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $bn\ \alpha\ \#* \ xvec3$ **by simp**
from $\langle A_P\ \#* \ (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $A_P\ \#* \ xvec3$ **and** $A_P\ \#* \ C$ **by simp+**

from $\langle length\ xvec3 = residualLength\ cRs \rangle\ C$ **have** $length\ xvec3 = length(bn\ \alpha)$
by simp
then obtain p **where** $S: set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **and** $distinctPerm\ p$ **and** $xvec3 = bn(p \cdot \alpha)$
using $\langle bn\ \alpha\ \#* \ xvec3 \rangle\ \langle distinct(bn\ \alpha) \rangle\ \langle distinct\ xvec3 \rangle$
by $- (rule\ constructPerm$ [**where** $xvec=bn\ \alpha$ **and** $yvec=xvec3$], *auto simp add: eqts*)
show *?thesis*
proof(*rule rPar2*[**where** $P=P$ **and** $Q=Q$ **and** $\alpha=p \cdot \alpha$ **and** $Q'=p \cdot Q'$ **and** $A_P=A_P$ **and** $\Psi_P=\Psi_P$])
assume $xvec3\ \#* \ \Psi$ **and** $xvec3\ \#* \ cP$ **and** $xvec3\ \#* \ cRs$
note $\langle cP = P \parallel Q \rangle$
moreover from $B\ C\ S\ \langle bn\ \alpha\ \#* \ xvec3 \rangle\ \langle xvec3\ \#* \ cRs \rangle\ \langle xvec3 = bn(p \cdot \alpha) \rangle\ \langle bn\ \alpha\ \#* \ subject\ \alpha \rangle\ \langle xvec3\ \#* \ cP \rangle\ \langle bn\ \alpha\ \#* \ P \rangle$


```

have cRs = (p · α) < P || (p · Q')
  apply clarsimp
  by(subst residualAlpha[where p=p]) auto
moreover note ⟨xvec3 = bn(p · α)⟩
moreover from ⟨Ψ ⊗ Ψ_P ▷ Q ⟶ α < Q'⟩ S B C S ⟨bn α #* xvec3⟩ ⟨xvec3
#* cRs⟩ ⟨xvec3 = bn(p · α)⟩ ⟨bn α #* subject α⟩ ⟨xvec3 #* cP⟩
have Ψ ⊗ Ψ_P ▷ Q ⟶ (p · α) < (p · Q')
  by(subst residualAlpha[symmetric]) auto
moreover note ⟨extractFrame P = ⟨A_P, Ψ_P⟩⟩ ⟨distinct A_P⟩ ⟨A_P #* P⟩ ⟨A_P
#* Q⟩ ⟨A_P #* Ψ⟩ ⟨A_P #* α⟩
moreover from ⟨A_P #* α⟩ ⟨A_P #* xvec3⟩ S ⟨xvec3 = bn(p · α)⟩ ⟨distinctPerm
p⟩ have A_P #* (p · α)
  by(subst fresh-star-bij[symmetric, where pi=p]) simp
moreover from ⟨A_P #* Q'⟩ ⟨A_P #* α⟩ ⟨A_P #* xvec3⟩ S ⟨xvec3 = bn(p · α)⟩
⟨distinctPerm p⟩ have A_P #* (p · Q')
  by(subst fresh-star-bij[symmetric, where pi=p]) simp
moreover note ⟨A_P #* C⟩
ultimately show cP = P || Q ∧ cRs = (p · α) < P || (p · Q') ∧ xvec3 = bn
(p · α) ∧
  Ψ ⊗ Ψ_P ▷ Q ⟶ (p · α) < p · Q' ∧
  extractFrame P = ⟨A_P, Ψ_P⟩ ∧ distinct A_P ∧ A_P #* P ∧ A_P #* Q ∧ A_P #*
Ψ ∧ A_P #* (p · α) ∧
  A_P #* (p · Q') ∧ A_P #* C by blast
qed
next
case(cComm1 Ψ_Q P M N P' A_P Ψ_P Q K xvec Q' A_Q)
  then show ?thesis by - (rule rComm1[where P=P and Q=Q], (assumption |
simp)+)
next
case(cComm2 Ψ_Q P M xvec N P' A_P Ψ_P Q K Q' A_Q)
  then show ?thesis by - (rule rComm2[where P=P and Q=Q], (assumption |
simp)+)
next
case(cBrMerge Ψ_Q P M N P' A_P Ψ_P Q Q' A_Q)
  then show ?thesis by - (rule rBrMerge[where P=P and Q=Q], (assumption
| simp)+)
next
case(cBrComm1 Ψ_Q P M N P' A_P Ψ_P Q xvec Q' A_Q)
  have B: cP = P || Q and C: cRs = iM(ν*xvec)⟨N⟩ < P' || Q'
  by fact+
  from ⟨xvec #* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,
x2, x3, x4, cP, cRs, C)⟩ have xvec #* xvec7 and xvec #* C by simp+
  from ⟨A_P #* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,
x2, x3, x4, cP, cRs, C)⟩ have A_P #* xvec7 and A_P #* C by simp+
  from ⟨A_Q #* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,
x2, x3, x4, cP, cRs, C)⟩ have A_Q #* xvec7 and A_Q #* C by simp+

from ⟨length xvec7 = residualLength cRs⟩ C have length xvec7 = length xvec
  by simp

```

then obtain p where $S: \text{set } p \subseteq \text{set } \text{vec} \times \text{set } (p \cdot \text{vec})$ and $\text{distinctPerm } p$
and $\text{vec7} = p \cdot \text{vec}$
using $\langle \text{vec} \#* \text{vec7} \rangle \langle \text{distinct vec} \rangle \langle \text{distinct vec7} \rangle$
by $-$ (rule $\text{constructPerm}[\text{where } \text{vec}=\text{vec} \text{ and } \text{yvec}=\text{vec7}]$, $\text{auto simp add: eqts}$)
show $?thesis$
proof (rule $\text{rBrComm1}[\text{where } P=P \text{ and } Q=Q \text{ and } P'=p \cdot P'$
and $Q'=p \cdot Q'$ and $N=p \cdot N$ and $A_P=A_P$ and $\Psi_P=\Psi_P$ and $A_Q=A_Q$
and $\Psi_Q=\Psi_Q$ and $M=M$])
assume $\text{vec7} \#* \Psi$ and $\text{vec7} \#* cP$ and $\text{vec7} \#* cRs$
from $\langle A_Q \#* \text{vec7} \rangle \langle \text{vec7} = p \cdot \text{vec} \rangle$
have $A_Q \#* (p \cdot \text{vec})$ by simp
from $\langle A_P \#* \text{vec7} \rangle \langle \text{vec7} = p \cdot \text{vec} \rangle$
have $A_P \#* (p \cdot \text{vec})$ by simp
from $\langle \text{vec} \#* M \rangle$ have $(p \cdot \text{vec}) \#* (p \cdot M)$ by ($\text{simp add: fresh-star-bij}$)
with $S \langle \text{vec} \#* M \rangle \langle \text{vec7} \#* cRs \rangle C \langle \text{vec7} = p \cdot \text{vec} \rangle$ have $\text{vec7} \#* M$ by
 simp

note $\langle cP = P \parallel Q \rangle$
moreover from $C S \langle \text{vec} \#* \text{vec7} \rangle \langle \text{vec7} \#* cRs \rangle \langle \text{vec7} = p \cdot \text{vec} \rangle \langle \text{vec} \#* M \rangle \langle \text{vec7} \#* cP \rangle$
have $cRs = \text{!}M(\nu*\text{vec7})\langle (p \cdot N) \rangle \prec (p \cdot P') \parallel (p \cdot Q')$
apply clarsimp
by ($\text{subst residualAlpha}[\text{where } p=p]$) simp+

moreover note $\langle \text{vec7} = p \cdot \text{vec} \rangle$
moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \text{!}M(\text{!}N) \prec P' \rangle S B \langle \text{distinctPerm } p \rangle \langle \text{vec} \#* \text{vec7} \rangle \langle \text{vec7} = p \cdot \text{vec} \rangle \langle \text{vec} \#* P \rangle \langle \text{vec7} \#* cP \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto \text{!}M(\text{!}(p \cdot N)) \prec p \cdot P'$
by ($\text{simp add: brinputAlpha}$)

moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{!}M(\text{!}\nu*\text{vec})\langle N \rangle \prec Q' \rangle S B C \langle \text{vec} \#* \text{vec7} \rangle \langle \text{vec7} \#* cRs \rangle \langle \text{vec7} = p \cdot \text{vec} \rangle \langle \text{vec} \#* M \rangle \langle \text{vec7} \#* cP \rangle \langle \text{vec7} \#* M \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto \text{!}M(\text{!}\nu*\text{vec7})\langle (p \cdot N) \rangle \prec p \cdot Q'$
by ($\text{auto simp add: residualAlpha}$)

moreover note
 $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle \text{distinct } A_P \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle \text{distinct } A_Q \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \text{vec7} \rangle$
 $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \text{vec7} \rangle$

moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* N \rangle$
have $A_P \#* (p \cdot N)$
by ($\text{simp add: freshChainSimps}$)
moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* P' \rangle$
have $A_P \#* (p \cdot P')$
by ($\text{simp add: freshChainSimps}$)
moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* Q' \rangle$

have $A_P \#* (p \cdot Q')$
by(*simp add: freshChainSimps*)
moreover from $S \langle A_Q \#* xvec \rangle \langle A_Q \#* (p \cdot xvec) \rangle \langle A_Q \#* N \rangle$
have $A_Q \#* (p \cdot N)$
by(*simp add: freshChainSimps*)
moreover from $S \langle A_Q \#* xvec \rangle \langle A_Q \#* (p \cdot xvec) \rangle \langle A_Q \#* P' \rangle$
have $A_Q \#* (p \cdot P')$
by(*simp add: freshChainSimps*)
moreover from $S \langle A_Q \#* xvec \rangle \langle A_Q \#* (p \cdot xvec) \rangle \langle A_Q \#* Q' \rangle$
have $A_Q \#* (p \cdot Q')$
by(*simp add: freshChainSimps*)

moreover note $\langle xvec7 \#* \Psi \rangle$

moreover from $\langle xvec7 \#* cP \rangle \langle cP = P \parallel Q \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle A_P \#* xvec7 \rangle$
have $xvec7 \#* \Psi_P$
by *simp (metis extractFrameFreshChain freshFrameDest)*

moreover from $\langle xvec7 \#* cP \rangle \langle cP = P \parallel Q \rangle \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_Q \#* xvec7 \rangle$
have $xvec7 \#* \Psi_Q$
by *simp (metis extractFrameFreshChain freshFrameDest)*

moreover from $\langle xvec7 \#* cP \rangle \langle cP = P \parallel Q \rangle$ **have** $xvec7 \#* P$ **by** *simp*
moreover from $\langle xvec7 \#* cP \rangle \langle cP = P \parallel Q \rangle$ **have** $xvec7 \#* Q$ **by** *simp*

moreover note $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle$

moreover note $\langle xvec7 \#* M \rangle$

moreover note $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle distinct xvec7 \rangle$

ultimately show $cP = P \parallel Q \wedge cRs = \downarrow M(\nu * xvec7) \langle (p \cdot N) \rangle \prec (p \cdot P') \parallel (p \cdot Q') \wedge$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\langle (p \cdot N) \rangle) \prec p \cdot P' \wedge extractFrame P = \langle A_P, \Psi_P \rangle \wedge$
distinct $A_P \wedge$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\nu * xvec7) \langle (p \cdot N) \rangle \prec p \cdot Q' \wedge extractFrame Q = \langle A_Q, \Psi_Q \rangle \wedge$
distinct $A_Q \wedge$
 $A_P \#* \Psi \wedge A_P \#* \Psi_Q \wedge A_P \#* P \wedge A_P \#* (p \cdot N) \wedge A_P \#* (p \cdot P') \wedge A_P \#*$
 $Q \wedge A_P \#* (p \cdot Q') \wedge$
 $A_P \#* A_Q \wedge A_P \#* xvec7 \wedge A_Q \#* \Psi \wedge A_Q \#* \Psi_P \wedge A_Q \#* P \wedge A_Q \#* (p \cdot$
 $N) \wedge A_Q \#* (p \cdot P') \wedge$
 $A_Q \#* Q \wedge A_Q \#* (p \cdot Q') \wedge A_Q \#* xvec7 \wedge xvec7 \#* \Psi \wedge xvec7 \#* \Psi_P \wedge$
 $xvec7 \#* \Psi_Q \wedge xvec7 \#* P \wedge$
 $xvec7 \#* Q \wedge A_P \#* M \wedge A_Q \#* M \wedge xvec7 \#* M \wedge A_P \#* C \wedge A_Q \#* C \wedge$
distinct $xvec7$ **by** *blast*

qed
next

case($cBrComm2 \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q$)
have $B: cP = P \parallel Q$ **and** $C: cRs = \downarrow M(\nu * xvec)(N) \prec P' \parallel Q'$
by *fact+*
from $\langle xvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $xvec \#* xvec8$ **and** $xvec \#* C$ **by** *simp+*
from $\langle A_P \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $A_P \#* xvec8$ **and** $A_P \#* C$ **by** *simp+*
from $\langle A_Q \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $A_Q \#* xvec8$ **and** $A_Q \#* C$ **by** *simp+*

from $\langle length \ xvec8 = residualLength \ cRs \rangle \ C$ **have** $length \ xvec8 = length \ xvec$
by *simp*
then obtain p **where** $S: set \ p \subseteq set \ xvec \times set \ (p \cdot xvec)$ **and** *distinctPerm* p
and $xvec8 = p \cdot xvec$
using $\langle xvec \#* xvec8 \rangle \ \langle distinct \ xvec \rangle \ \langle distinct \ xvec8 \rangle$
by $- (rule \ constructPerm[where \ xvec=xvec \ \text{and} \ yvec=xvec8], \ auto \ simp \ add: \ eqts)$
show *?thesis*
proof(*rule* $rBrComm2[where \ P=P \ \text{and} \ Q=Q \ \text{and} \ P'=p \cdot P'$
and $Q'=p \cdot Q' \ \text{and} \ N=p \cdot N \ \text{and} \ A_P=A_P \ \text{and} \ \Psi_P=\Psi_P \ \text{and} \ A_Q=A_Q$
and $\Psi_Q=\Psi_Q \ \text{and} \ M=M]$)
assume $xvec8 \#* \Psi$ **and** $xvec8 \#* cP$ **and** $xvec8 \#* cRs$
from $\langle A_Q \#* xvec8 \rangle \ \langle xvec8 = p \cdot xvec \rangle$
have $A_Q \#* (p \cdot xvec)$ **by** *simp*
from $\langle A_P \#* xvec8 \rangle \ \langle xvec8 = p \cdot xvec \rangle$
have $A_P \#* (p \cdot xvec)$ **by** *simp*
from $\langle xvec \#* M \rangle$ **have** $(p \cdot xvec) \#* (p \cdot M)$ **by**(*simp add: fresh-star-bij*)
with $S \ \langle xvec \#* M \rangle \ \langle xvec8 \#* cRs \rangle \ C \ \langle xvec8 = p \cdot xvec \rangle$ **have** $xvec8 \#* M$ **by**
simp

note $\langle cP = P \parallel Q \rangle$
moreover from $C \ S \ \langle xvec \#* xvec8 \rangle \ \langle xvec8 \#* cRs \rangle \ \langle xvec8 = p \cdot xvec \rangle \ \langle xvec \#* M \rangle \ \langle xvec8 \#* cP \rangle$
have $cRs = \downarrow M(\nu * xvec8)(p \cdot N) \prec (p \cdot P') \parallel (p \cdot Q')$
apply *clarsimp*
by(*subst residualAlpha[where p=p]*) *simp+*

moreover note $\langle xvec8 = p \cdot xvec \rangle$
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\downarrow N) \prec Q' \rangle \ S \ B \ \langle distinctPerm \ p \rangle \ \langle xvec \#* xvec8 \rangle \ \langle xvec8 = p \cdot xvec \rangle \ \langle xvec \#* Q \rangle \ \langle xvec8 \#* cP \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\downarrow (p \cdot N)) \prec p \cdot Q'$
by(*simp add: brinputAlpha*)

moreover from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\nu * xvec)(N) \prec P' \rangle \ S \ B \ C \ \langle xvec \#* xvec8 \rangle \ \langle xvec8 \#* cRs \rangle \ \langle xvec8 = p \cdot xvec \rangle \ \langle xvec \#* M \rangle \ \langle xvec8 \#* cP \rangle \ \langle xvec8 \#* M \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\nu * xvec8)(p \cdot N) \prec p \cdot P'$
by(*auto simp add: residualAlpha*)

moreover note

$\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle \text{distinct } A_P \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle \text{distinct } A_Q \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \text{vec8} \rangle$
 $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \text{vec8} \rangle$

moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* N \rangle$
have $A_P \#* (p \cdot N)$
by (*simp add: freshChainSimps*)

moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* P' \rangle$
have $A_P \#* (p \cdot P')$
by (*simp add: freshChainSimps*)

moreover from $S \langle A_P \#* \text{vec} \rangle \langle A_P \#* (p \cdot \text{vec}) \rangle \langle A_P \#* Q' \rangle$
have $A_P \#* (p \cdot Q')$
by (*simp add: freshChainSimps*)

moreover from $S \langle A_Q \#* \text{vec} \rangle \langle A_Q \#* (p \cdot \text{vec}) \rangle \langle A_Q \#* N \rangle$
have $A_Q \#* (p \cdot N)$
by (*simp add: freshChainSimps*)

moreover from $S \langle A_Q \#* \text{vec} \rangle \langle A_Q \#* (p \cdot \text{vec}) \rangle \langle A_Q \#* P' \rangle$
have $A_Q \#* (p \cdot P')$
by (*simp add: freshChainSimps*)

moreover from $S \langle A_Q \#* \text{vec} \rangle \langle A_Q \#* (p \cdot \text{vec}) \rangle \langle A_Q \#* Q' \rangle$
have $A_Q \#* (p \cdot Q')$
by (*simp add: freshChainSimps*)

moreover note $\langle \text{vec8} \#* \Psi \rangle$

moreover from $\langle \text{vec8} \#* cP \rangle \langle cP = P \parallel Q \rangle \langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle A_P \#* \text{vec8} \rangle$
have $\text{vec8} \#* \Psi_P$
by *simp (metis extractFrameFreshChain freshFrameDest)*

moreover from $\langle \text{vec8} \#* cP \rangle \langle cP = P \parallel Q \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_Q \#* \text{vec8} \rangle$
have $\text{vec8} \#* \Psi_Q$
by *simp (metis extractFrameFreshChain freshFrameDest)*

moreover from $\langle \text{vec8} \#* cP \rangle \langle cP = P \parallel Q \rangle$ **have** $\text{vec8} \#* P$ **by** *simp*
moreover from $\langle \text{vec8} \#* cP \rangle \langle cP = P \parallel Q \rangle$ **have** $\text{vec8} \#* Q$ **by** *simp*

moreover note $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle$

moreover note $\langle \text{vec8} \#* M \rangle$

moreover note $\langle A_P \#* C \rangle \langle A_Q \#* C \rangle \langle \text{distinct } \text{vec8} \rangle$

ultimately show $cP = P \parallel Q \wedge cRs = \text{jM}(\nu * \text{vec8}) \langle (p \cdot N) \rangle \prec (p \cdot P') \parallel (p \cdot Q') \wedge$
 $\Psi \otimes \Psi_Q \triangleright P \mapsto \text{jM}(\nu * \text{vec8}) \langle (p \cdot N) \rangle \prec p \cdot P' \wedge \text{extractFrame } P = \langle A_P, \Psi_P \rangle \wedge \text{distinct } A_P \wedge$

```

     $\Psi \otimes \Psi_P \triangleright Q \mapsto iM((p \cdot N)) \prec p \cdot Q' \wedge \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \wedge$ 
    distinct  $A_Q \wedge$ 
     $A_P \#* \Psi \wedge A_P \#* \Psi_Q \wedge A_P \#* P \wedge A_P \#* (p \cdot N) \wedge A_P \#* (p \cdot P') \wedge A_P \#*$ 
 $Q \wedge A_P \#* (p \cdot Q') \wedge$ 
     $A_P \#* A_Q \wedge A_P \#* \text{vec8} \wedge A_Q \#* \Psi \wedge A_Q \#* \Psi_P \wedge A_Q \#* P \wedge A_Q \#* (p \cdot$ 
 $N) \wedge A_Q \#* (p \cdot P') \wedge$ 
     $A_Q \#* Q \wedge A_Q \#* (p \cdot Q') \wedge A_Q \#* \text{vec8} \wedge \text{vec8} \#* \Psi \wedge \text{vec8} \#* \Psi_P \wedge$ 
 $\text{vec8} \#* \Psi_Q \wedge \text{vec8} \#* P \wedge$ 
     $\text{vec8} \#* Q \wedge A_P \#* M \wedge A_Q \#* M \wedge \text{vec8} \#* M \wedge A_P \#* C \wedge A_Q \#* C \wedge$ 
distinct  $\text{vec8}$  by blast
  qed
next
  case(cBrClose  $P M \text{vec} N P' x$ )
  have  $B: cP = (\nu x)P$  and  $C: cRs = \tau \prec (\nu x)((\nu * \text{vec})P')$ 
  by fact+
  from  $\langle x \# (\text{vec1}, \text{vec2}, \text{vec3}, \text{vec4}, \text{vec5}, \text{vec6}, \text{vec7}, \text{vec8}, \text{vec9}, x1, x2,$ 
 $x3, x4, cP, cRs, C) \rangle$  have  $x \# cP$  and  $x \# cRs$  and  $x \neq x3$  by simp+
  from  $\langle \text{vec} \#* (\text{vec1}, \text{vec2}, \text{vec3}, \text{vec4}, \text{vec5}, \text{vec6}, \text{vec7}, \text{vec8}, \text{vec9}, x1,$ 
 $x2, x3, x4, cP, cRs, C) \rangle$  have  $\text{vec} \#* cP$  and  $\text{vec} \#* cRs$  and  $x3 \# \text{vec}$  and
 $\text{vec} \#* x3$  by simp+

  obtain  $r::\text{name prm}$  where  $(r \cdot \text{vec}) \#* \Psi$  and  $(r \cdot \text{vec}) \#* P$  and  $(r \cdot \text{vec})$ 
 $\#* M$ 
    and  $(r \cdot \text{vec}) \#* N$  and  $(r \cdot \text{vec}) \#* P'$  and  $(r \cdot \text{vec}) \#* x$ 
    and  $(r \cdot \text{vec}) \#* C$  and  $(r \cdot \text{vec}) \#* x3$ 
    and  $(r \cdot \text{vec}) \#* ((x, x3) \cdot P)$  and  $(r \cdot \text{vec}) \#* ((x, x3) \cdot M)$ 
    and  $Sr: (\text{set } r) \subseteq (\text{set } \text{vec}) \times (\text{set}(r \cdot \text{vec}))$  and distinctPerm  $r$ 
    by(rule name-list-avoiding[where  $\text{vec}=\text{vec}$  and  $c=(\Psi, P, M, N, P', x, x3,$ 
 $((x, x3) \cdot P), ((x, x3) \cdot P), C)]$ )
    (auto simp add: eqvts)
    from  $\langle (r \cdot \text{vec}) \#* x3 \rangle$  have  $x3 \# (r \cdot \text{vec})$  by simp

  show ?thesis
  proof(rule rBrClose[where  $P=[(x, x3)] \cdot P$  and  $M=[(x, x3)] \cdot M$  and  $N=[(x,$ 
 $x3)] \cdot (r \cdot N)$  and  $\text{vec}=(r \cdot \text{vec})$  and  $P'=[(x, x3)] \cdot (r \cdot P')$ )
    assume  $x3 \# \Psi$  and  $x3 \# cP$  and  $x3 \# cRs$ 
    with  $\langle x3 \# \text{vec} \rangle$  have  $x3 \notin \text{set } \text{vec}$  by simp

    from  $\langle x3 \# cRs \rangle$   $C$  have  $x3 \# (\tau \prec (\nu x)((\nu * \text{vec})P'))$  by simp

    then have  $x3 \# ((\nu x)((\nu * \text{vec})P'))$  by simp
    with  $\langle x \neq x3 \rangle$  have  $x3 \# ((\nu * \text{vec})P')$  by(simp add: psi.fresh abs-fresh)
    then have  $(x3 \in \text{set } \text{vec}) \vee (x3 \# P')$  by(simp add: resChainFresh)

    with  $\langle x3 \notin \text{set } \text{vec} \rangle$  have  $x3 \# P'$  by blast

    with  $\langle x3 \# \text{vec} \rangle \langle x3 \# (r \cdot \text{vec}) \rangle$   $Sr$  have  $x3 \# (r \cdot P')$ 
    by(simp add: freshChainSimps)

```

from $\langle cP = (\nu x)P \rangle \langle x\exists \# cP \rangle \langle x \neq x\exists \rangle$ **have** $cP\text{-perm}: cP = (\nu x\exists)(([x, x\exists]) \cdot P)$
by (*simp add: alphaRes abs-fresh*)
from $\langle (r \cdot xvec) \#* P' \rangle Sr C$
have $cRs = \tau \prec (\nu x)((\nu*(r \cdot xvec))(r \cdot P'))$
by (*simp add: resChainAlpha*)
moreover from $\langle x\exists \# (r \cdot P') \rangle$
have $x\exists \# (\nu*(r \cdot xvec))(r \cdot P')$
by (*simp add: resChainFresh*)
ultimately have $cRs = \tau \prec (\nu x\exists)(([x, x\exists]) \cdot (\nu*(r \cdot xvec))(r \cdot P'))$ **by** (*simp add: alphaRes*)
with $\langle (r \cdot xvec) \#* x \rangle \langle (r \cdot xvec) \#* x\exists \rangle$
have $cRs\text{-perm}: cRs = \tau \prec (\nu x\exists)((\nu*(r \cdot xvec))([x, x\exists]) \cdot (r \cdot P'))$
by (*simp add: eqvts*)

from $\langle x \in \text{supp } M \rangle$
have $\text{supp-inc-perm}: x\exists \in \text{supp } ([x, x\exists]) \cdot M$
by (*metis fresh-bij fresh-def swap-simps*)

from $\langle x \# xvec \rangle \langle (r \cdot xvec) \#* x \rangle Sr$ **have** $r \cdot x = x$ **by** *simp*

from $\langle \Psi \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P' \rangle \langle (r \cdot xvec) \#* N \rangle \langle (r \cdot xvec) \#* P' \rangle Sr$
have $\Psi \triangleright P \mapsto iM(\nu*(r \cdot xvec))\langle (r \cdot N) \rangle \prec (r \cdot P')$
by (*simp add: boundOutputChainAlpha'' create-residual.simps*)
then have $[[x, x\exists]] \cdot (\Psi \triangleright P \mapsto iM(\nu*(r \cdot xvec))\langle (r \cdot N) \rangle \prec (r \cdot P'))$
by (*simp add: perm-bool*)
with $\langle x \# \Psi \rangle \langle x\exists \# \Psi \rangle \langle (r \cdot xvec) \#* x \rangle \langle x\exists \# (r \cdot xvec) \rangle$
have $\text{trans-perm}: \Psi \triangleright ([[x, x\exists]] \cdot P) \mapsto i([[x, x\exists]] \cdot M)(\nu*(r \cdot xvec))\langle ([[x, x\exists]] \cdot (r \cdot N)) \rangle \prec ([[x, x\exists]] \cdot (r \cdot P'))$
by (*auto simp add: eqvts*)

from $\langle \text{distinctPerm } r \rangle \langle \text{distinct } xvec \rangle$
have $\text{distinct-perm}: \text{distinct } (r \cdot xvec)$ **by** *simp*

note $cP\text{-perm } cRs\text{-perm } \text{supp-inc-perm } \text{trans-perm } \text{distinct-perm}$
 $\langle (r \cdot xvec) \#* \Psi \rangle \langle (r \cdot xvec) \#* ([[x, x\exists]] \cdot P) \rangle \langle (r \cdot xvec) \#* ([[x, x\exists]] \cdot M) \rangle$
 $\langle (r \cdot xvec) \#* C \rangle \langle x\exists \# \Psi \rangle \langle x\exists \# (r \cdot xvec) \rangle$

then show $cP = (\nu x\exists)(([x, x\exists]) \cdot P) \wedge cRs = \tau \prec (\nu x\exists)((\nu*(r \cdot xvec))([x, x\exists]) \cdot r \cdot P') \wedge$
 $x\exists \in \text{supp } ([x, x\exists]) \cdot M \wedge$
 $\Psi \triangleright [[x, x\exists]] \cdot P \mapsto i([[x, x\exists]] \cdot M)(\nu*(r \cdot xvec))\langle ([[x, x\exists]] \cdot r \cdot N) \rangle \prec [[x, x\exists]] \cdot r \cdot P' \wedge$
 $\text{distinct } (r \cdot xvec) \wedge (r \cdot xvec) \#* \Psi \wedge (r \cdot xvec) \#* ([[x, x\exists]] \cdot P) \wedge$
 $(r \cdot xvec) \#* ([[x, x\exists]] \cdot M) \wedge (r \cdot xvec) \#* C \wedge x\exists \# \Psi \wedge x\exists \# r \cdot xvec$
by *simp*

qed
next
case(*cOpen P M xvec yvec N P' x*)

have $B: cP = (\nu x)P$ **and** $C: cRs = M(\nu*(xvec@x\#yvec))\langle N \rangle \prec P'$
by *fact+*
from $\langle xvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $xvec \#* xvec4$ **and** $xvec \#* cP$ **and** $xvec \#* cRs$ **and** $x1 \# xvec$ **by** *simp+*
from $\langle x \# (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $x \# xvec4$ **and** $x \# cP$ **and** $x \# cRs$ **and** $x \neq x1$ **by** *simp+*
from $\langle yvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $yvec \#* xvec4$ **and** $yvec \#* cP$ **and** $yvec \#* cRs$ **and** $x1 \# yvec$ **by** *simp+*

from $\langle xvec \#* cRs \rangle \langle x \# cRs \rangle \langle yvec \#* cRs \rangle C$ **have** $(xvec@x\#yvec) \#* M$ **by** *simp*
from $\langle xvec \#* \Psi \rangle \langle x \# \Psi \rangle \langle yvec \#* \Psi \rangle$ **have** $(xvec@x\#yvec) \#* \Psi$ **by** *simp*
from $\langle length\ xvec4 = residualLength\ cRs \rangle C$ **obtain** $xvec' y yvec'$ **where** $D: xvec4 = xvec'@y\#yvec'$ **and** $length\ xvec' = length\ xvec$ **and** $length\ yvec' = length\ yvec$
by(*auto intro: lengthAux2*)
with $\langle distinct\ xvec \rangle \langle distinct\ yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle xvec \#* yvec \rangle \langle xvec \#* xvec4 \rangle \langle yvec \#* xvec4 \rangle \langle x \# xvec4 \rangle \langle distinct\ xvec4 \rangle$
have $distinct\ xvec'$ **and** $distinct\ yvec'$ **and** $xvec' \#* yvec'$ **and** $x \neq y$ **and** $y \# xvec'$ **and** $y \# yvec'$
and $x \# xvec'$ **and** $x \# yvec'$ **and** $y \# xvec$ **and** $y \# yvec$ **and** $xvec \#* xvec'$ **and** $yvec \#* yvec'$
by *auto*
from $\langle length\ xvec' = length\ xvec \rangle \langle xvec \#* xvec' \rangle \langle distinct\ xvec \rangle \langle distinct\ xvec' \rangle$
obtain p **where** $Sp: set\ p \subseteq set\ xvec \times set(p \cdot xvec)$ **and** $distinctPerm\ p$ **and** $E: xvec' = p \cdot xvec$
by(*metis constructPerm*)
from $\langle length\ yvec' = length\ yvec \rangle \langle yvec \#* yvec' \rangle \langle distinct\ yvec \rangle \langle distinct\ yvec' \rangle$
obtain q **where** $Sq: set\ q \subseteq set\ yvec \times set(q \cdot yvec)$ **and** $distinctPerm\ q$ **and** $F: yvec' = q \cdot yvec$
by(*metis constructPerm*)

show *?thesis*
proof(*rule rOpen[where P=([(x, x1)] \cdot P) and xvec=p \cdot xvec and y=y and yvec=q \cdot yvec and N=(p@(x1, x)\#q) \cdot N and P'=(p@(x1, x)\#q) \cdot P' and M=M]*)
assume $xvec4 \#* \Psi$ **and** $xvec4 \#* cP$ **and** $xvec4 \#* cRs$ **and** $x1 \# \Psi$ **and** $x1 \# cP$ **and** $x1 \# cRs$ **and** $x1 \# xvec4$
from $\langle xvec \#* xvec4 \rangle \langle x \# xvec4 \rangle \langle x1 \# xvec4 \rangle \langle yvec \#* xvec4 \rangle D E F$
have $x \neq y$ **and** $x1 \neq y$ **and** $x1 \# p \cdot xvec$ **and** $x1 \# q \cdot yvec$ **by** *simp+*
from $\langle xvec4 \#* cRs \rangle \langle x1 \# cRs \rangle C$ **have** $xvec4 \#* M$ **and** $x1 \# M$ **by** *simp+*
moreover **from** $\langle cP = (\nu x)P \rangle \langle x \# cP \rangle \langle x \neq x1 \rangle$ **have** $([(x, x1)] \cdot cP) = [(x, x1)] \cdot (\nu x)P$
by *simp*
with $\langle x \# cP \rangle \langle x1 \# cP \rangle$ **have** $cP = (\nu x1)(([x, x1]] \cdot P)$ **by**(*simp add: eqts calc-atm*)

moreover from C **have** $((p@x1, x)\#q) \cdot cRs = (p@(x1, x)\#q) \cdot (M(\nu*(xvec@x\#yvec))\langle N \rangle \prec P')$ **by** (*simp add: fresh-star-bij*)

with $Sp Sq \langle xvec4 \#* cRs \rangle D E F \langle xvec \#* cRs \rangle \langle x \# cRs \rangle \langle yvec \#* cRs \rangle \langle xvec4 \#* M \rangle \langle (xvec@x\#yvec) \#* M \rangle \langle xvec \#* xvec4 \rangle \langle x \# xvec4 \rangle \langle yvec \#* xvec4 \rangle \langle xvec \#* yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle y \# xvec' \rangle \langle y \# yvec' \rangle \langle xvec' \#* yvec' \rangle \langle x1 \# xvec \rangle \langle x1 \# yvec \rangle \langle x1 \neq y \rangle \langle x1 \# xvec4 \rangle \langle x1 \# cRs \rangle \langle x1 \# cRs \rangle \langle x \neq x1 \rangle \langle x1 \# M \rangle$

have $cRs = M(\nu*((p \cdot xvec)@x1\#(q \cdot yvec)))\langle ((p@(x1, x)\#q) \cdot N) \rangle \prec ((p@(x1, x)\#q) \cdot P')$

by (*simp add: eqvts pt2[OF pt-name-inst] calc-atm*)

moreover from $D E F$ **have** $xvec4 = (p \cdot xvec)@y\#(q \cdot yvec)$ **by** *simp*

moreover from $\langle \Psi \triangleright P \mapsto M(\nu*(xvec@yvec))\langle N \rangle \prec P' \rangle$ **have** $((p@(x1, x)\#q) \cdot \Psi) \triangleright ((p@(x1, x)\#q) \cdot P) \mapsto ((p@(x1, x)\#q) \cdot (M(\nu*(xvec@yvec))\langle N \rangle \prec P'))$

by (*intro eqvts*)

with $Sp Sq B C D E F \langle xvec4 \#* \Psi \rangle \langle (xvec@x\#yvec) \#* \Psi \rangle \langle xvec4 \#* cRs \rangle \langle x \# xvec4 \rangle C D \langle x \# cRs \rangle \langle yvec \#* cRs \rangle \langle xvec4 \#* M \rangle \langle (xvec@x\#yvec) \#* M \rangle \langle x \# M \rangle \langle x1 \# cRs \rangle \langle x \neq x1 \rangle \langle x1 \# xvec \rangle \langle x1 \# yvec \rangle \langle xvec \#* xvec4 \rangle \langle yvec \#* xvec4 \rangle \langle x1 \# xvec4 \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x1 \# \Psi \rangle \langle xvec4 \#* cP \rangle \langle xvec \#* P \rangle \langle yvec \#* P \rangle \langle xvec' \#* yvec' \rangle \langle x1 \# xvec4 \rangle \langle xvec4 \#* cP \rangle \langle yvec \#* xvec4 \rangle \langle xvec \#* xvec4 \rangle \langle x \neq x1 \rangle \langle xvec \#* yvec \rangle$

have $\Psi \triangleright (([x, x1]) \cdot P) \mapsto M(\nu*((p \cdot xvec)@y\#(q \cdot yvec)))\langle ((p@(x1, x)\#q) \cdot N) \rangle \prec ((p@(x1, x)\#q) \cdot P')$

by (*simp add: eqvts pt-fresh-bij[OF pt-name-inst, OF at-name-inst] pt2[OF pt-name-inst] name-swap*)

moreover from $\langle x \in \text{supp } N \rangle$ **have** $((p@(x1, x)\#q) \cdot x) \in ((p@(x1, x)\#q) \cdot \text{supp } N)$

by (*simp add: pt-set-bij[OF pt-name-inst, OF at-name-inst]*)

then have $x1 \in \text{supp}((p@(x1, x)\#q) \cdot N)$

using $\langle x \# xvec \rangle \langle x \# yvec \rangle \langle x1 \# xvec \rangle \langle x1 \# yvec \rangle \langle x \# xvec4 \rangle \langle x1 \# xvec4 \rangle \langle xvec \#* xvec4 \rangle \langle yvec \#* xvec4 \rangle \langle xvec' \#* yvec' \rangle D E F Sp Sq \langle x \neq x1 \rangle$

by (*simp add: eqvts pt2[OF pt-name-inst] calc-atm*)

moreover from $\langle x1 \# xvec4 \rangle D E F$ **have** $x1 \# (p \cdot xvec)$ **and** $x1 \# (q \cdot yvec)$

by *simp+*

moreover from $\langle \text{distinct } xvec' \rangle \langle \text{distinct } yvec' \rangle E F$ **have** $\text{distinct}(p \cdot xvec)$

and $\text{distinct}(q \cdot yvec)$ **by** *simp+*

moreover from $\langle xvec' \#* yvec' \rangle E F$ **have** $(p \cdot xvec) \#* (q \cdot yvec)$ **by** *auto*

moreover from $\langle xvec \#* \Psi \rangle$ **have** $(p \cdot xvec) \#* (p \cdot \Psi)$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $Sp D E \langle xvec4 \#* \Psi \rangle \langle xvec \#* \Psi \rangle$ **have** $(p \cdot xvec) \#* \Psi$ **by** (*simp add: eqvts*)

moreover from $\langle yvec \#* \Psi \rangle$ **have** $(p \cdot yvec) \#* (p \cdot \Psi)$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $Sq D F \langle xvec4 \#* \Psi \rangle \langle yvec \#* \Psi \rangle$ **have** $(q \cdot yvec) \#* \Psi$ **by** (*simp add: eqvts*)

moreover from $\langle xvec4 \#* cP \rangle \langle x \# xvec4 \rangle \langle x1 \# xvec4 \rangle B D E F$ **have** $(p \cdot xvec) \#* ([x, x1]) \cdot P$ **and** $(q \cdot yvec) \#* ([x, x1]) \cdot P$

by *simp+*

moreover from $\langle xvec4 \#* M \rangle C D E F$ **have** $(p \cdot xvec) \#* M$ **and** $(q \cdot yvec)$

$\#* M$ **by simp+**
ultimately show $cP = (\nu x1) \langle [(x, x1)] \cdot P \rangle \wedge$
 $cRs = M(\nu*(p \cdot xvec @ x1 \# q \cdot yvec)) \langle ((p @ (x1, x) \# q) \cdot N) \rangle \prec (p @$
 $(x1, x) \# q) \cdot P' \wedge$
 $xvec4 = p \cdot xvec @ y \# q \cdot yvec \wedge$
 $\Psi \triangleright [(x, x1)] \cdot P \mapsto M(\nu*(p \cdot xvec @ q \cdot yvec)) \langle ((p @ (x1, x) \# q) \cdot N) \rangle$
 $\prec (p @ (x1, x) \# q) \cdot P' \wedge$
 $x1 \in \text{supp} ((p @ (x1, x) \# q) \cdot N) \wedge$
 $x1 \# p \cdot xvec \wedge$
 $x1 \# q \cdot yvec \wedge$
 $\text{distinct } (p \cdot xvec) \wedge$
 $\text{distinct } (q \cdot yvec) \wedge$
 $(p \cdot xvec) \#* \Psi \wedge$
 $(p \cdot xvec) \#* [(x, x1)] \cdot P \wedge$
 $(p \cdot xvec) \#* M \wedge$
 $(p \cdot xvec) \#* (q \cdot yvec) \wedge$
 $(q \cdot yvec) \#* \Psi \wedge$
 $(q \cdot yvec) \#* [(x, x1)] \cdot P \wedge$
 $(q \cdot yvec) \#* M$
by blast
qed
next
case($cBrOpen P M xvec yvec N P' x$)
have $B: cP = (\nu x)P$ **and** $C: cRs = \imath M(\nu*(xvec @ x \# yvec)) \langle N \rangle \prec P'$
by fact+
from $\langle xvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9,$
 $x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $xvec \#* xvec9$ **and** $xvec \#* cP$ **and** $xvec \#* cRs$
and $x4 \# xvec$ **by simp+**
from $\langle x \# (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2,$
 $x3, x4, cP, cRs, C) \rangle$ **have** $x \# xvec9$ **and** $x \# cP$ **and** $x \# cRs$ **and** $x \neq x4$ **by**
 simp+
from $\langle yvec \#* (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1,$
 $x2, x3, x4, cP, cRs, C) \rangle$ **have** $yvec \#* xvec9$ **and** $yvec \#* cP$ **and** $yvec \#* cRs$
and $x4 \# yvec$ **by simp+**

from $\langle xvec \#* cRs \rangle \langle x \# cRs \rangle \langle yvec \#* cRs \rangle C$ **have** $(xvec @ x \# yvec) \#* M$ **by**
 simp
from $\langle xvec \#* \Psi \rangle \langle x \# \Psi \rangle \langle yvec \#* \Psi \rangle$ **have** $(xvec @ x \# yvec) \#* \Psi$ **by simp**
from $\langle \text{length } xvec9 = \text{residualLength } cRs \rangle C$ **obtain** $xvec' y yvec'$ **where** $D:$
 $xvec9 = xvec' @ y \# yvec'$ **and** $\text{length } xvec' = \text{length } xvec$ **and** $\text{length } yvec' = \text{length}$
 $yvec$
by($\text{auto intro: lengthAux2}$)
with $\langle \text{distinct } xvec \rangle \langle \text{distinct } yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle xvec \#* yvec \rangle \langle xvec \#*$
 $xvec9 \rangle \langle yvec \#* xvec9 \rangle \langle x \# xvec9 \rangle \langle \text{distinct } xvec9 \rangle$
have $\text{distinct } xvec'$ **and** $\text{distinct } yvec'$ **and** $xvec' \#* yvec'$ **and** $x \neq y$ **and** $y \#$
 $xvec'$ **and** $y \# yvec'$
and $x \# xvec'$ **and** $x \# yvec'$ **and** $y \# xvec$ **and** $y \# yvec$ **and** $xvec \#* xvec'$ **and**
 $yvec \#* yvec'$
by auto

from $\langle \text{length } xvec' = \text{length } xvec \rangle \langle xvec \#* xvec' \rangle \langle \text{distinct } xvec \rangle \langle \text{distinct } xvec' \rangle$
obtain p **where** $Sp: \text{set } p \subseteq \text{set } xvec \times \text{set}(p \cdot xvec)$ **and** $\text{distinctPerm } p$ **and**
 $E: xvec' = p \cdot xvec$
by(metis constructPerm)
from $\langle \text{length } yvec' = \text{length } yvec \rangle \langle yvec \#* yvec' \rangle \langle \text{distinct } yvec \rangle \langle \text{distinct } yvec' \rangle$
obtain q **where** $Sq: \text{set } q \subseteq \text{set } yvec \times \text{set}(q \cdot yvec)$ **and** $\text{distinctPerm } q$ **and**
 $F: yvec' = q \cdot yvec$
by(metis constructPerm)

show ?thesis

proof(rule rBrOpen[**where** $P = ([x, x_4]) \cdot P$) **and** $xvec = p \cdot xvec$ **and** $y = y$
and $yvec = q \cdot yvec$ **and** $N = (p@(x_4, x)\#q) \cdot N$ **and** $P' = (p@(x_4, x)\#q) \cdot P'$ **and**
 $M = M$])

assume $xvec9 \#* \Psi$ **and** $xvec9 \#* cP$ **and** $xvec9 \#* cRs$ **and** $x_4 \# \Psi$ **and** $x_4 \#$
 cP **and** $x_4 \# cRs$ **and** $x_4 \# xvec9$

from $\langle xvec \#* xvec9 \rangle \langle x \# xvec9 \rangle \langle x_4 \# xvec9 \rangle \langle yvec \#* xvec9 \rangle D E F$

have $x \neq y$ **and** $x_4 \neq y$ **and** $x_4 \# p \cdot xvec$ **and** $x_4 \# q \cdot yvec$ **by** simp+

from $\langle xvec9 \#* cRs \rangle \langle x_4 \# cRs \rangle C$ **have** $xvec9 \#* M$ **and** $x_4 \# M$ **by** simp+

moreover from $\langle cP = (\nu x)P \rangle \langle x \# cP \rangle \langle x \neq x_4 \rangle$ **have** $([x, x_4]) \cdot cP = ([x,$
 $x_4]) \cdot (\nu x)P$

by simp

with $\langle x \# cP \rangle \langle x_4 \# cP \rangle$ **have** $cP = (\nu x_4)([x, x_4]) \cdot P$ **by**(simp add: eqvts
calc-atm)

moreover from C **have** $((p@(x_4, x)\#q) \cdot cRs) = (p@(x_4, x)\#q) \cdot (iM(\nu*(xvec@x\#yvec))\langle N \rangle$
 $\prec P')$ **by**(simp add: fresh-star-bij)

with $Sp Sq \langle xvec9 \#* cRs \rangle D E F \langle xvec \#* cRs \rangle \langle x \# cRs \rangle \langle yvec \#* cRs \rangle \langle xvec9$
 $\#* M \rangle \langle (xvec@x\#yvec) \#* M \rangle \langle xvec \#* xvec9 \rangle \langle x \# xvec9 \rangle \langle yvec \#* xvec9 \rangle \langle xvec \#*$
 $yvec \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle y \# yvec' \rangle \langle y \# yvec' \rangle \langle xvec' \#* yvec' \rangle \langle x_4 \# xvec \rangle \langle x_4 \#$
 $yvec \rangle \langle x_4 \neq y \rangle \langle x_4 \# xvec9 \rangle \langle x_4 \# cRs \rangle \langle x_4 \# cRs \rangle \langle x \neq x_4 \rangle \langle x_4 \# M \rangle$

have $cRs = iM(\nu*((p \cdot xvec)@x_4\#(q \cdot yvec)))\langle ((p@(x_4, x)\#q) \cdot N) \rangle \prec$
 $((p@(x_4, x)\#q) \cdot P')$

by(simp add: eqvts pt2[OF pt-name-inst] calc-atm)

moreover from $D E F$ **have** $xvec9 = (p \cdot xvec)@y\#(q \cdot yvec)$ **by** simp

moreover from $\langle \Psi \triangleright P \mapsto iM(\nu*(xvec@yvec))\langle N \rangle \prec P' \rangle$ **have** $((p@(x_4,$
 $x)\#q) \cdot \Psi) \triangleright ((p@(x_4, x)\#q) \cdot P) \mapsto ((p@(x_4, x)\#q) \cdot (iM(\nu*(xvec@yvec))\langle N \rangle$
 $\prec P'))$

by(intro eqvts)

with $Sp Sq B C D E F \langle xvec9 \#* \Psi \rangle \langle (xvec@x\#yvec) \#* \Psi \rangle \langle xvec9 \#* cRs \rangle$
 $\langle x \# xvec9 \rangle C D \langle x \# cRs \rangle \langle yvec \#* cRs \rangle \langle xvec9 \#* M \rangle \langle (xvec@x\#yvec) \#* M \rangle \langle x$
 $\# M \rangle \langle x_4 \# cRs \rangle \langle x \neq x_4 \rangle \langle x_4 \# xvec \rangle \langle x_4 \# yvec \rangle \langle xvec \#* xvec9 \rangle \langle yvec \#* xvec9 \rangle$
 $\langle x_4 \# xvec9 \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x_4 \# \Psi \rangle \langle xvec9 \#* cP \rangle \langle xvec \#* P \rangle \langle yvec \#* P \rangle$
 $\langle xvec' \#* yvec' \rangle \langle x_4 \# xvec9 \rangle \langle xvec9 \#* cP \rangle \langle yvec \#* xvec9 \rangle \langle xvec \#* xvec9 \rangle \langle x \neq$
 $x_4 \rangle \langle xvec \#* yvec \rangle$

have $\Psi \triangleright ([x, x_4]) \cdot P \mapsto iM(\nu*((p \cdot xvec)@y\#(q \cdot yvec)))\langle ((p@(x_4, x)\#q) \cdot$
 $N) \rangle \prec ((p@(x_4, x)\#q) \cdot P')$

by(simp add: eqvts pt-fresh-bij[OF pt-name-inst, OF at-name-inst] pt2[OF
pt-name-inst] name-swap)

moreover from $\langle x \in \text{supp } N \rangle$ **have** $((p@(x_4, x)\#q) \cdot x) \in ((p@(x_4, x)\#q) \cdot$

$\text{supp } N$
by(*simp add: pt-set-bij*[*OF pt-name-inst, OF at-name-inst*])
then have $x_4 \in \text{supp}((p @ (x_4, x) \# q) \cdot N)$
using $\langle x \# \text{xvec} \rangle \langle x \# \text{yvec} \rangle \langle x_4 \# \text{xvec} \rangle \langle x_4 \# \text{yvec} \rangle \langle x \# \text{xvec9} \rangle \langle x_4 \# \text{xvec9} \rangle$
 $\langle \text{xvec} \#* \text{xvec9} \rangle \langle \text{yvec} \#* \text{xvec9} \rangle \langle \text{xvec}' \#* \text{yvec}' \rangle D E F Sp Sq \langle x \neq x_4 \rangle$
by(*simp add: eqts pt2*[*OF pt-name-inst*] *calc-atm*)
moreover from $\langle x_4 \# \text{xvec9} \rangle D E F$ **have** $x_4 \# (p \cdot \text{xvec})$ **and** $x_4 \# (q \cdot \text{yvec})$
by *simp+*
moreover from $\langle \text{distinct } \text{xvec}' \rangle \langle \text{distinct } \text{yvec}' \rangle E F$ **have** $\text{distinct}(p \cdot \text{xvec})$
and $\text{distinct}(q \cdot \text{yvec})$ **by** *simp+*
moreover from $\langle \text{xvec}' \#* \text{yvec}' \rangle E F$ **have** $(p \cdot \text{xvec}) \#* (q \cdot \text{yvec})$ **by** *auto*
moreover from $\langle \text{xvec} \#* \Psi \rangle$ **have** $(p \cdot \text{xvec}) \#* (p \cdot \Psi)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $Sp D E \langle \text{xvec9} \#* \Psi \rangle \langle \text{xvec} \#* \Psi \rangle$ **have** $(p \cdot \text{xvec}) \#* \Psi$ **by**(*simp add: eqts*)
moreover from $\langle \text{yvec} \#* \Psi \rangle$ **have** $(p \cdot \text{yvec}) \#* (p \cdot \Psi)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $Sq D F \langle \text{xvec9} \#* \Psi \rangle \langle \text{yvec} \#* \Psi \rangle$ **have** $(q \cdot \text{yvec}) \#* \Psi$ **by**(*simp add: eqts*)
moreover from $\langle \text{xvec9} \#* cP \rangle \langle x \# \text{xvec9} \rangle \langle x_4 \# \text{xvec9} \rangle B D E F$ **have** $(p \cdot \text{xvec}) \#* ([x, x_4]) \cdot P$ **and** $(q \cdot \text{yvec}) \#* ([x, x_4]) \cdot P$
by *simp+*
moreover from $\langle \text{xvec9} \#* M \rangle C D E F$ **have** $(p \cdot \text{xvec}) \#* M$ **and** $(q \cdot \text{yvec}) \#* M$ **by** *simp+*
ultimately show $cP = (\nu x_4)([x, x_4]) \cdot P \wedge$
 $cRs = \text{iM}(\nu*(p \cdot \text{xvec} @ x_4 \# q \cdot \text{yvec}))(((p @ (x_4, x) \# q) \cdot N)) \prec (p @ (x_4, x) \# q) \cdot P' \wedge$
 $\text{xvec9} = p \cdot \text{xvec} @ y \# q \cdot \text{yvec} \wedge$
 $\Psi \triangleright [(x, x_4)] \cdot P \longmapsto \text{iM}(\nu*(p \cdot \text{xvec} @ q \cdot \text{yvec}))(((p @ (x_4, x) \# q) \cdot N))$
 $\prec (p @ (x_4, x) \# q) \cdot P' \wedge$
 $x_4 \in \text{supp}((p @ (x_4, x) \# q) \cdot N) \wedge$
 $x_4 \# p \cdot \text{xvec} \wedge$
 $x_4 \# q \cdot \text{yvec} \wedge$
 $\text{distinct}(p \cdot \text{xvec}) \wedge$
 $\text{distinct}(q \cdot \text{yvec}) \wedge$
 $(p \cdot \text{xvec}) \#* \Psi \wedge$
 $(p \cdot \text{xvec}) \#* ([x, x_4]) \cdot P \wedge$
 $(p \cdot \text{xvec}) \#* M \wedge$
 $(p \cdot \text{xvec}) \#* (q \cdot \text{yvec}) \wedge$
 $(q \cdot \text{yvec}) \#* \Psi \wedge$
 $(q \cdot \text{yvec}) \#* ([x, x_4]) \cdot P \wedge$
 $(q \cdot \text{yvec}) \#* M$
by *blast*
qed
next
case(*cScope P* α $P' x$)
have $B: cP = (\nu x)P$ **and** $C: cRs = \alpha \prec (\nu x)P'$
by *fact+*
from $\langle \text{bn } \alpha \#* (\text{xvec1}, \text{xvec2}, \text{xvec3}, \text{xvec4}, \text{xvec5}, \text{xvec6}, \text{xvec7}, \text{xvec8}, \text{xvec9})$

$x1, x2, x3, x4, cP, cRs, C$ have $bn \alpha \#* xvec5$ and $x2 \# bn \alpha$ by *simp+*
from $\langle x \# (xvec1, xvec2, xvec3, xvec4, xvec5, xvec6, xvec7, xvec8, xvec9, x1, x2, x3, x4, cP, cRs, C) \rangle$ **have** $x \# xvec5$ and $x \neq x2$ and $x \# cRs$ by *simp+*

from $\langle length \ xvec5 = residualLength \ cRs \rangle \ C$ **have** $length \ xvec5 = length(bn \ \alpha)$
by *simp*
then obtain p where $S: set \ p \subseteq set(bn \ \alpha) \times set(bn(p \cdot \alpha))$ and *distinctPerm*
 p and $xvec5 = bn(p \cdot \alpha)$
using $\langle bn \ \alpha \#* xvec5 \rangle \ \langle distinct(bn \ \alpha) \rangle \ \langle distinct \ xvec5 \rangle$
by $-$ (*rule constructPerm*[**where** $xvec = bn \ \alpha$ and $yvec = xvec5$], *auto simp add:*
eqts)
show *?thesis*
proof(*rule rScope*[**where** $P = [(x, x2)] \cdot P$ and $\alpha = [(x, x2)] \cdot p \cdot \alpha$ and $P' = [(x, x2)] \cdot p \cdot P'$])
assume $xvec5 \#* \Psi$ and $xvec5 \#* cP$ and $xvec5 \#* cRs$ and $x2 \# \Psi$ and $x2 \#$
 cP and $x2 \# cRs$ and $x2 \# xvec5$
from $\langle x2 \# cRs \rangle \ C \ \langle x2 \# bn \ \alpha \rangle \ \langle x \neq x2 \rangle$ **have** $x2 \# \alpha$ and $x2 \# P'$ by(*auto*
simp add: abs-fresh)
moreover from $\langle cP = (\nu x)P \rangle \ \langle x2 \# cP \rangle \ \langle x \neq x2 \rangle$ **have** $cP = (\nu x2)([(x, x2)]$
 $\cdot P)$
by(*simp add: alphaRes abs-fresh*)
moreover from $B \ C \ S \ \langle bn \ \alpha \#* xvec5 \rangle \ \langle xvec5 \#* cRs \rangle \ \langle xvec5 = bn(p \cdot \alpha) \rangle$
 $\langle bn \ \alpha \#* subject \ \alpha \rangle \ \langle xvec5 \#* cP \rangle \ \langle x \# \alpha \rangle \ \langle x \# xvec5 \rangle$
have $cRs = (p \cdot \alpha) \prec (\nu x)(p \cdot P')$
apply *clarsimp*
by(*subst residualAlpha*[**where** $p = p$] *alphaRes*) (*auto simp del: actionFresh*)
then have $([(x, x2)] \cdot cRs) = [(x, x2)] \cdot ((p \cdot \alpha) \prec (\nu x)(p \cdot P'))$
by *simp*
with $\langle x2 \# cRs \rangle \ \langle x \# cRs \rangle$ **have** $cRs = ([x, x2] \cdot p \cdot \alpha) \prec (\nu x2)([x, x2] \cdot$
 $p \cdot P')$
by(*simp add: eqts calc-atm*)
moreover from $\langle xvec5 = bn(p \cdot \alpha) \rangle$ **have** $([x, x2] \cdot xvec5) = ([x, x2] \cdot bn(p$
 $\cdot \alpha))$
by *simp*
with $\langle x \# xvec5 \rangle \ \langle x2 \# xvec5 \rangle$ **have** $xvec5 = bn([x, x2] \cdot p \cdot \alpha)$
by(*simp add: eqts*)
moreover from $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \ S \ B \ C \ S \ \langle bn \ \alpha \#* xvec5 \rangle \ \langle xvec5 \#* cRs \rangle$
 $\langle xvec5 = bn(p \cdot \alpha) \rangle \ \langle bn \ \alpha \#* subject \ \alpha \rangle \ \langle xvec5 \#* cP \rangle \ \langle x \# xvec5 \rangle$
have $\Psi \triangleright P \mapsto (p \cdot \alpha) \prec (p \cdot P')$
by(*subst residualAlpha*[*symmetric*] *auto*)
then have $([x, x2] \cdot \Psi) \triangleright ([x, x2] \cdot P) \mapsto ([x, x2] \cdot ((p \cdot \alpha) \prec (p \cdot P')))$
by(*rule eqvt*)
with $\langle x \# \Psi \rangle \ \langle x2 \# \Psi \rangle$ **have** $\Psi \triangleright ([x, x2] \cdot P) \mapsto ([x, x2] \cdot p \cdot \alpha) \prec ([x,$
 $x2] \cdot p \cdot P')$
by(*simp add: eqts*)
moreover note $\langle x2 \# \Psi \rangle$
moreover from $\langle x \# \alpha \rangle \ \langle x2 \# \alpha \rangle \ \langle x \# xvec5 \rangle \ \langle x2 \# xvec5 \rangle \ S \ \langle x \neq x2 \rangle \ \langle xvec5$
 $= bn(p \cdot \alpha) \rangle$ **have** $x2 \# [(x, x2)] \cdot p \cdot \alpha$
apply(*subgoal-tac* $x \# p \wedge x2 \# p$)

```

    apply(simp add: perm-compose freshChainSimps del: actionFresh)
  by(auto dest: freshAlphaSwap)
  moreover from ⟨bn α #* subject α⟩ have ([[x, x2]] · p · (bn α)) #* ([[x, x2]]
· p · (subject α))
    by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  then have bn([[x, x2]] · p · α) #* subject([[x, x2]] · p · α)
    by(simp add: eqvts)
  moreover from ⟨distinct(bn α)⟩ have distinct([[x, x2]] · p · (bn α)) by simp
  then have distinct(bn([[x, x2]] · p · α)) by(simp add: eqvts)
  ultimately show cP = (νx2)([[x, x2]] · P) ∧
    cRs = ([[x, x2]] · p · α) < (νx2)([[x, x2]] · p · P') ∧
    xvec5 = bn ([[x, x2]] · p · α) ∧
    Ψ ▷ [[x, x2]] · P ⟶ ([[x, x2]] · p · α) < [[x, x2]] · p · P' ∧
    x2 # Ψ ∧
    x2 # [[x, x2]] · p · α ∧
    bn ([[x, x2]] · p · α) #* subject ([[x, x2]] · p · α) ∧
    distinct (bn ([[x, x2]] · p · α)) by blast
qed
next
case(cBang P)
  then show ?thesis by(auto intro: rBang)
qed

```

lemma *resResidEq*:

```

  fixes xvec :: name list
  and P     :: ('a, 'b, 'c) psi
  and Q     :: ('a, 'b, 'c) psi
  and M     :: 'a
  and N     :: 'a
  and N'    :: 'a

```

```

assumes iM(ν*xvec)⟨N⟩ < P = iM(ν*xvec)⟨N'⟩ < Q
  and xvec #* M

```

```

shows iM(ν*xvec)⟨N⟩ = iM(ν*xvec)⟨N'⟩

```

```

  using assms

```

```

proof(induct xvec)

```

```

  case Nil

```

```

    then show ?case by(simp add: residualInject)

```

```

next

```

```

  case (Cons x xvec)

```

```

  from ⟨x # xvec⟩ #* M

```

```

  have x # M and xvec #* M by simp+

```

```

  from ⟨iM(ν*(x # xvec))⟨N⟩ < P = iM(ν*(x # xvec))⟨N'⟩ < Q⟩ ⟨x # M⟩

```

```

  have iM(ν*(xvec))⟨N⟩ < P = iM(ν*(xvec))⟨N'⟩ < Q

```

```

  by (metis action.inject(4) assms(1) bn.simps(4) residualInject'')

```

```

  then have iM(ν*xvec)⟨N⟩ = iM(ν*xvec)⟨N'⟩ using ⟨xvec #* M⟩

```

```

  by(rule Cons(1))

```

```

  then show ?case

```

by(*simp add: action.inject*)
qed

lemma *parCases*[*consumes 5, case-names cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2*]:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and Q :: ('a, 'b, 'c) psi
and α :: 'a action
and T :: ('a, 'b, 'c) psi
and C :: 'f::fs-name
and M :: 'a
and N :: 'a

assumes *Trans*: $\Psi \triangleright P \parallel Q \mapsto \alpha \prec T$

and $bn \ \alpha \ \#* \ \Psi$
and $bn \ \alpha \ \#* \ P$
and $bn \ \alpha \ \#* \ Q$
and $bn \ \alpha \ \#* \ subject \ \alpha$
and *rPar1*: $\bigwedge P' A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'; \ extractFrame \ Q = \langle A_Q, \Psi_Q \rangle; \ distinct \ A_Q;$

$A_Q \ \#* \ \Psi; A_Q \ \#* \ P; A_Q \ \#* \ Q; A_Q \ \#* \ \alpha; A_Q \ \#* \ P'; A_Q \ \#* \ C \rrbracket \implies Prop \ \alpha \ (P' \parallel Q)$

and *rPar2*: $\bigwedge Q' A_P \Psi_P. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'; \ extractFrame \ P = \langle A_P, \Psi_P \rangle; \ distinct \ A_P;$

$A_P \ \#* \ \Psi; A_P \ \#* \ P; A_P \ \#* \ Q; A_P \ \#* \ \alpha; A_P \ \#* \ Q'; A_P \ \#* \ C \rrbracket \implies Prop \ \alpha \ (P \parallel Q')$

and *rComm1*: $\bigwedge \Psi_Q M N P' A_P \Psi_P K \ xvec \ Q' A_Q.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; \ extractFrame \ P = \langle A_P, \Psi_P \rangle; \ distinct \ A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec) \langle N \rangle \prec Q'; \ extractFrame \ Q = \langle A_Q, \Psi_Q \rangle;$

distinct $A_Q;$

$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; \ distinct \ xvec; \ \alpha = \tau;$

$A_P \ \#* \ \Psi; A_P \ \#* \ \Psi_Q; A_P \ \#* \ P; A_P \ \#* \ M; A_P \ \#* \ N; A_P \ \#* \ P'; A_P \ \#* \ Q; A_P \ \#* \ xvec; A_P \ \#* \ Q'; A_P \ \#* \ A_Q; A_P \ \#* \ C;$

$A_Q \ \#* \ \Psi; A_Q \ \#* \ \Psi_P; A_Q \ \#* \ P; A_Q \ \#* \ K; A_Q \ \#* \ N; A_Q \ \#* \ P'; A_Q \ \#* \ Q; A_Q \ \#* \ xvec; A_Q \ \#* \ Q'; A_Q \ \#* \ C;$

$xvec \ \#* \ \Psi; xvec \ \#* \ \Psi_P; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ K; xvec \ \#* \ Q; xvec \ \#* \ \Psi_Q; xvec \ \#* \ C \rrbracket \implies$

$Prop \ (\tau) \ ((\nu * xvec) \langle P' \parallel Q' \rangle)$

and *rComm2*: $\bigwedge \Psi_Q M \ xvec \ N P' A_P \Psi_P K Q' A_Q.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'; \ extractFrame \ P = \langle A_P, \Psi_P \rangle;$

distinct $A_P;$

$\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; \ extractFrame \ Q = \langle A_Q, \Psi_Q \rangle; \ distinct \ A_Q;$

$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; \ distinct \ xvec; \ \alpha = \tau;$

$A_P \ \#* \ \Psi; A_P \ \#* \ \Psi_Q; A_P \ \#* \ P; A_P \ \#* \ M; A_P \ \#* \ N; A_P \ \#* \ P'; A_P \ \#* \ Q; A_P \ \#* \ xvec; A_P \ \#* \ Q'; A_P \ \#* \ A_Q; A_P \ \#* \ C;$

$A_Q \ \#* \ \Psi; A_Q \ \#* \ \Psi_P; A_Q \ \#* \ P; A_Q \ \#* \ K; A_Q \ \#* \ N; A_Q \ \#* \ P'; A_Q \ \#* \ Q; A_Q \ \#* \ xvec; A_Q \ \#* \ Q'; A_Q \ \#* \ C;$

$xvec \ \#* \ \Psi; xvec \ \#* \ \Psi_P; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ K; xvec \ \#* \ Q;$

$xvec \#* \Psi_Q; xvec \#* C \implies$
 $Prop (\tau) (\nu*xvec)(P' \parallel Q')$
and $rBrMerge: \bigwedge \Psi_Q M N P' A_P \Psi_P Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
distinct $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct $A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C; \alpha = \iota M(N) \implies$
 $Prop (\iota M(N)) (P' \parallel Q')$
and $rBrComm1: \bigwedge \Psi_Q M N P' A_P \Psi_P xvec Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct$
 $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\nu*xvec)(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct $A_Q;$
distinct $xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#*$
 $xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#*$
 $xvec; A_Q \#* Q'; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M; \iota M(\nu*xvec)(N) = \alpha;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q \implies$
 $Prop (\iota M(\nu*xvec)(N)) (P' \parallel Q')$
and $rBrComm2: \bigwedge \Psi_Q M xvec N P' A_P \Psi_P Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(\nu*xvec)(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
distinct $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct$
 $A_Q;$
distinct $xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#*$
 $xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#*$
 $xvec; A_Q \#* Q'; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M; \iota M(\nu*xvec)(N) = \alpha;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q \implies$
 $Prop (\iota M(\nu*xvec)(N)) (P' \parallel Q')$

shows $Prop \alpha T$

proof –

from $Trans$ **have** $distinct(bn \alpha)$ **by** $(auto dest: boundOutputDistinct)$

have $length(bn \alpha) = residualLength(\alpha \prec T)$ **by** $simp$

note $Trans$

moreover have $length [] = inputLength(P \parallel Q)$ **and** $distinct []$

by $(auto simp add: inputLength-inputLength'-inputLength''.simps)$

moreover have $length [] = inputLength(P \parallel Q)$ **and** $distinct []$

by $(auto simp add: inputLength-inputLength'-inputLength''.simps)$

moreover note $\langle length(bn \alpha) = residualLength(\alpha \prec T) \rangle \langle distinct(bn \alpha) \rangle$


```

moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover note  $\langle \text{length}(bn \ \alpha) = \text{residualLength}(\alpha \prec T) \rangle \langle \text{distinct}(bn \ \alpha) \rangle$ 
moreover obtain  $x::\text{name}$  where  $x \# \Psi$  and  $x \# P$  and  $x \# Q$  and  $x \# \alpha$  and
 $x \# T$ 
  by(generate-fresh name) auto
ultimately show ?thesis using  $\langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* P \rangle \langle bn \ \alpha \#* Q \rangle \langle bn \ \alpha \#*$ 
subject  $\alpha \rangle$ 
proof(cases rule: semanticsCases[of - - - - -  $C \ x \ x \ x$ ])
  case cInput
  then show ?thesis
    by(simp add: residualInject)
  next
  case cBrInput
  then show ?thesis
    by(simp add: residualInject)
  next
  case cOutput
  then show ?thesis
    by(simp add: residualInject)
  next
  case cBrOutput
  then show ?thesis
    by(simp add: residualInject)
  next
  case cCase
  then show ?thesis
    by(simp add: residualInject)
  next
  case (cPar1  $\Psi_Q \ P \ \alpha' \ P' \ Q \ A_Q$ )
  then show ?thesis using assms
    by(force simp add: psi.inject residualInject residualInject' intro: rPar1)
  next
  case (cPar2  $\Psi_P \ Q \ \alpha' \ Q' \ P \ A_P$ )
  then show ?thesis using assms
    by(force simp add: psi.inject residualInject residualInject' intro: rPar1)
  next
  case cComm1
  then show ?thesis using assms
    by(force simp add: psi.inject residualInject residualInject' intro: rComm1)
  next
  case cComm2
  then show ?thesis using assms
    by(force simp add: psi.inject residualInject residualInject' intro: rComm2)
  next
  case cBrMerge

```

then show *?thesis using assms*
by(*force simp add: psi.inject residualInject residualInject' intro: rBrMerge*)
next
case (*cBrComm1* Ψ_Q $P1$ M N P' A_P Ψ_P $Q1$ Q' A_Q)
note $\langle bn \ \alpha \ \#* \ \Psi \rangle$
moreover from $\langle bn \ \alpha \ \#* \ P \rangle$ $\langle bn \ \alpha \ \#* \ Q \rangle$ **have** $bn \ \alpha \ \#* \ (P \parallel Q)$ **by** *simp*
moreover from $\langle bn \ \alpha \ \#* \ \text{subject } \alpha \rangle$ **have** $bn \ \alpha \ \#* \ (\alpha \prec T)$ **by** *simp*
ultimately have all:
 $P \parallel Q = P1 \parallel Q1 \wedge$
 $\alpha \prec T = \text{iM}(\nu * bn \ \alpha) \langle N \rangle \prec P' \parallel Q' \wedge$
 $\Psi \otimes \Psi_Q \triangleright P1 \mapsto \text{iM} \langle N \rangle \prec P' \wedge$
 $\text{extractFrame } P1 = \langle A_P, \Psi_P \rangle \wedge$
 $\text{distinct } A_P \wedge$
 $\Psi \otimes \Psi_P \triangleright Q1 \mapsto \text{iM}(\nu * bn \ \alpha) \langle N \rangle \prec Q' \wedge$
 $\text{extractFrame } Q1 = \langle A_Q, \Psi_Q \rangle \wedge$
 $\text{distinct } A_Q \wedge$
 $A_P \ \#* \ \Psi \wedge$
 $A_P \ \#* \ \Psi_Q \wedge$
 $A_P \ \#* \ P1 \wedge$
 $A_P \ \#* \ N \wedge$
 $A_P \ \#* \ P' \wedge$
 $A_P \ \#* \ Q1 \wedge$
 $A_P \ \#* \ Q' \wedge$
 $A_P \ \#* \ A_Q \wedge$
 $A_P \ \#* \ bn \ \alpha \wedge$
 $A_Q \ \#* \ \Psi \wedge$
 $A_Q \ \#* \ \Psi_P \wedge$
 $A_Q \ \#* \ P1 \wedge$
 $A_Q \ \#* \ N \wedge$
 $A_Q \ \#* \ P' \wedge$
 $A_Q \ \#* \ Q1 \wedge$
 $A_Q \ \#* \ Q' \wedge$
 $A_Q \ \#* \ bn \ \alpha \wedge$
 $bn \ \alpha \ \#* \ \Psi \wedge$
 $bn \ \alpha \ \#* \ \Psi_P \wedge$
 $bn \ \alpha \ \#* \ \Psi_Q \wedge$
 $bn \ \alpha \ \#* \ P1 \wedge$
 $bn \ \alpha \ \#* \ Q1 \wedge$
 $A_P \ \#* \ M \wedge A_Q \ \#* \ M \wedge bn \ \alpha \ \#* \ M \wedge A_P \ \#* \ C \wedge A_Q \ \#* \ C \wedge \text{distinct } (bn \ \alpha)$
by(*rule cBrComm1(1)*)

from all have $bn \ \alpha \ \#* \ M$ **and** $\alpha \prec T = \text{iM}(\nu * bn \ \alpha) \langle N \rangle \prec P' \parallel Q'$
by *simp+*

from $\langle \alpha \prec T = \text{iM}(\nu * bn \ \alpha) \langle N \rangle \prec P' \parallel Q' \rangle$ **have** $\alpha \prec T = \text{RBrOut } M \ ((\nu * bn \ \alpha) N \prec' (P' \parallel Q'))$
by(*simp add: residualInject*)

then obtain *xvec* N' **where** $\alpha = \text{iM}(\nu * xvec) \langle N' \rangle$

by(*auto simp add: residualInject*)
then have $bn\ \alpha = xvec$ **by** *simp*
from $\langle \alpha = \mathit{jM}(\nu*xvec)\langle N' \rangle \ \langle \alpha \prec T = \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle \prec P' \parallel Q' \rangle$ **have**
resEq: $\mathit{jM}(\nu*xvec)\langle N' \rangle \prec T = \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle \prec P' \parallel Q'$
by *simp*
then have $\mathit{jM}(\nu*bn\ \alpha)\langle N' \rangle \prec T = \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle \prec P' \parallel Q'$ **using** $\langle bn\ \alpha = xvec \rangle$
by *simp*
then have $\mathit{jM}(\nu*bn\ \alpha)\langle N' \rangle = \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle$ **using** $\langle bn\ \alpha \#* M \rangle$
by(*rule resResidEq*)
with $\langle \alpha = \mathit{jM}(\nu*xvec)\langle N' \rangle \rangle$ **have** $\alpha = \mathit{jM}(\nu*xvec)\langle N \rangle$ **by** *simp*

moreover from all have $P \parallel Q = P1 \parallel Q1$
and $\alpha \prec T = \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle \prec P' \parallel Q'$
and $\Psi \otimes \Psi_Q \triangleright P1 \mapsto \mathit{jM}\langle N \rangle \prec P'$
and *extractFrame* $P1 = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \triangleright Q1 \mapsto \mathit{jM}(\nu*bn\ \alpha)\langle N \rangle \prec Q'$
and *extractFrame* $Q1 = \langle A_Q, \Psi_Q \rangle$
and *distinct* A_Q
and $A_P \#* \Psi$
and $A_P \#* \Psi_Q$
and $A_P \#* P1$
and $A_P \#* N$
and $A_P \#* P'$
and $A_P \#* Q1$
and $A_P \#* Q'$
and $A_P \#* A_Q$
and $A_P \#* bn\ \alpha$
and $A_Q \#* \Psi$
and $A_Q \#* \Psi_P$
and $A_Q \#* P1$
and $A_Q \#* N$
and $A_Q \#* P'$
and $A_Q \#* Q1$
and $A_Q \#* Q'$
and $A_Q \#* bn\ \alpha$
and $bn\ \alpha \#* \Psi$
and $bn\ \alpha \#* \Psi_P$
and $bn\ \alpha \#* \Psi_Q$
and $bn\ \alpha \#* P1$
and $bn\ \alpha \#* Q1$
and $A_P \#* M$
and $A_Q \#* M$
and $bn\ \alpha \#* M$
and $A_P \#* C$
and $A_Q \#* C$
and *distinct* $(bn\ \alpha)$
by *auto*

moreover then have $P = P1$ and $Q = Q1$

by(auto simp add: psi.inject)

ultimately have $\text{Prop } (\text{iM}(\nu^*(bn \ \alpha))\langle N \rangle) (P' \parallel Q')$

by(force intro: rBrComm1)

then show $?thesis$ using $\langle \alpha \prec T = \text{iM}(\nu^*bn \ \alpha)\langle N \rangle \prec P' \parallel Q' \rangle$ [symmetric]

by(force simp add: residualInject)

next

case (cBrComm2 $\Psi_Q \ P1 \ M \ N \ P' \ A_P \ \Psi_P \ Q1 \ Q' \ A_Q$)

note $\langle bn \ \alpha \ \#* \ \Psi \rangle$

moreover from $\langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle$ have $bn \ \alpha \ \#* (P \parallel Q)$ by simp

moreover from $\langle bn \ \alpha \ \#* \ \text{subject } \alpha \rangle$ have $bn \ \alpha \ \#* (\alpha \prec T)$ by simp

ultimately have all:

$P \parallel Q = P1 \parallel Q1 \wedge$

$\alpha \prec T = \text{iM}(\nu^*bn \ \alpha)\langle N \rangle \prec P' \parallel Q' \wedge$

$\Psi \otimes \Psi_Q \triangleright P1 \mapsto \text{iM}(\nu^*bn \ \alpha)\langle N \rangle \prec P' \wedge$

$\text{extractFrame } P1 = \langle A_P, \Psi_P \rangle \wedge$

$\text{distinct } A_P \wedge$

$\Psi \otimes \Psi_P \triangleright Q1 \mapsto \text{iM}(\nu^*N) \prec Q' \wedge$

$\text{extractFrame } Q1 = \langle A_Q, \Psi_Q \rangle \wedge$

$\text{distinct } A_Q \wedge$

$A_P \ \#* \ \Psi \wedge$

$A_P \ \#* \ \Psi_Q \wedge$

$A_P \ \#* \ P1 \wedge$

$A_P \ \#* \ N \wedge$

$A_P \ \#* \ P' \wedge$

$A_P \ \#* \ Q1 \wedge$

$A_P \ \#* \ Q' \wedge$

$A_P \ \#* \ A_Q \wedge$

$A_P \ \#* \ bn \ \alpha \wedge$

$A_Q \ \#* \ \Psi \wedge$

$A_Q \ \#* \ \Psi_P \wedge$

$A_Q \ \#* \ P1 \wedge$

$A_Q \ \#* \ N \wedge$

$A_Q \ \#* \ P' \wedge$

$A_Q \ \#* \ Q1 \wedge$

$A_Q \ \#* \ Q' \wedge$

$A_Q \ \#* \ bn \ \alpha \wedge$

$bn \ \alpha \ \#* \ \Psi \wedge$

$bn \ \alpha \ \#* \ \Psi_P \wedge$

$bn \ \alpha \ \#* \ \Psi_Q \wedge$

$bn \ \alpha \ \#* \ P1 \wedge$

$bn \ \alpha \ \#* \ Q1 \wedge$

$A_P \ \#* \ M \wedge A_Q \ \#* \ M \wedge bn \ \alpha \ \#* \ M \wedge A_P \ \#* \ C \wedge A_Q \ \#* \ C \wedge \text{distinct } (bn \ \alpha)$

by(rule cBrComm2(1))

from all have $bn \ \alpha \ \#* \ M$ and $\alpha \prec T = \text{iM}(\nu^*bn \ \alpha)\langle N \rangle \prec P' \parallel Q'$

by simp+

from $\langle \alpha \prec T = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P' \parallel Q' \rangle$ **have** $\alpha \prec T = \text{RBrOut } M (\langle \nu * \text{bn } \alpha \rangle N \prec' (P' \parallel Q'))$
by (*simp add: residualInject*)

then obtain $\text{vec } N'$ **where** $\alpha = \text{iM}(\nu * \text{vec}) \langle N' \rangle$
by (*auto simp add: residualInject*)

then have $\text{bn } \alpha = \text{vec}$ **by** *simp*

from $\langle \alpha = \text{iM}(\nu * \text{vec}) \langle N' \rangle \langle \alpha \prec T = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P' \parallel Q' \rangle$ **have**
 $\text{resEq: } \text{iM}(\nu * \text{vec}) \langle N' \rangle \prec T = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P' \parallel Q'$
by *simp*

then have $\text{iM}(\nu * \text{bn } \alpha) \langle N' \rangle \prec T = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P' \parallel Q'$ **using** $\langle \text{bn } \alpha = \text{vec} \rangle$
by *simp*

then have $\text{iM}(\nu * \text{bn } \alpha) \langle N' \rangle = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle$ **using** $\langle \text{bn } \alpha \#* M \rangle$
by (*rule resResidEq*)

with $\langle \alpha = \text{iM}(\nu * \text{vec}) \langle N' \rangle \rangle$ **have** $\alpha = \text{iM}(\nu * \text{vec}) \langle N \rangle$ **by** *simp*

moreover from all have $P \parallel Q = P1 \parallel Q1$
and $\alpha \prec T = \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P' \parallel Q'$
and $\Psi \otimes \Psi_Q \triangleright P1 \mapsto \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec P'$
and $\text{extractFrame } P1 = \langle A_P, \Psi_P \rangle$
and $\text{distinct } A_P$
and $\Psi \otimes \Psi_P \triangleright Q1 \mapsto \text{iM}(\nu * \text{bn } \alpha) \langle N \rangle \prec Q'$
and $\text{extractFrame } Q1 = \langle A_Q, \Psi_Q \rangle$
and $\text{distinct } A_Q$
and $A_P \#* \Psi$
and $A_P \#* \Psi_Q$
and $A_P \#* P1$
and $A_P \#* N$
and $A_P \#* P'$
and $A_P \#* Q1$
and $A_P \#* Q'$
and $A_P \#* A_Q$
and $A_P \#* \text{bn } \alpha$
and $A_Q \#* \Psi$
and $A_Q \#* \Psi_P$
and $A_Q \#* P1$
and $A_Q \#* N$
and $A_Q \#* P'$
and $A_Q \#* Q1$
and $A_Q \#* Q'$
and $A_Q \#* \text{bn } \alpha$
and $\text{bn } \alpha \#* \Psi$
and $\text{bn } \alpha \#* \Psi_P$
and $\text{bn } \alpha \#* \Psi_Q$
and $\text{bn } \alpha \#* P1$
and $\text{bn } \alpha \#* Q1$
and $A_P \#* M$
and $A_Q \#* M$

```

and bn α #* M
and AP #* C
and AQ #* C
and distinct (bn α)
by auto

moreover then have P = P1 and Q = Q1
by(auto simp add: psi.inject)

ultimately have Prop (jM(ν*(bn α))(N)) (P' || Q')
by(force intro: rBrComm2)
then show ?thesis using ⟨α < T = jM(ν*bn α)(N) < P' || Q'⟩[symmetric]
by(force simp add: residualInject)
next
case cBrClose
then show ?thesis using ⟨x # Ψ⟩ ⟨x # P⟩ ⟨x # Q⟩ ⟨x # α⟩ ⟨x # T⟩
by(simp add: residualInject)
next
case cOpen
then show ?thesis using assms ⟨x # Ψ⟩ ⟨x # P⟩ ⟨x # Q⟩ ⟨x # α⟩ ⟨x # T⟩
by(simp add: residualInject)
next
case cBrOpen
then show ?thesis using assms ⟨x # Ψ⟩ ⟨x # P⟩ ⟨x # Q⟩ ⟨x # α⟩ ⟨x # T⟩
by(simp add: residualInject)
next
case cScope
then show ?thesis using assms ⟨x # Ψ⟩ ⟨x # P⟩ ⟨x # Q⟩ ⟨x # α⟩ ⟨x # T⟩
by(simp add: residualInject)
next
case cBang
then show ?thesis
by(simp add: residualInject)
qed
qed

lemma parInputCases[consumes 1, case-names cPar1 cPar2]:
fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and Q :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a
and R :: ('a, 'b, 'c) psi
and C :: 'f::fs-name

assumes Trans: Ψ ▷ P || Q ⟶ M(N) < R
and rPar1: ⋀P' AQ ΨQ. [[Ψ ⊗ ΨQ ▷ P ⟶ M(N) < P'; extractFrame Q =
⟨AQ, ΨQ⟩; distinct AQ;
AQ #* Ψ; AQ #* P; AQ #* Q; AQ #* M; AQ #* N; AQ #* C]]

```

$\implies Prop (P' \parallel Q)$
and $rPar2: \bigwedge Q' A_P \Psi_P. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(N) \prec Q' \rrbracket; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_P \#* N; A_P \#* C$
 $\implies Prop (P \parallel Q')$
shows $Prop R$
proof –
from $Trans$ **obtain** α **where** $\Psi \triangleright P \parallel Q \mapsto \alpha \prec R$ **and** $bn \alpha \#* \Psi$ **and** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ **and** $bn \alpha \#*$ *subject* α **and** $\alpha = M(N)$ **by** *auto*
then show *?thesis using rPar1 rPar2*
by(*induct rule: parCases*) (*auto simp add: residualInject*)
qed

lemma $parBrInputCases[consumes 1, case-names cPar1 cPar2 cBrMerge]:$

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $Q :: ('a, 'b, 'c) psi$
and $M :: 'a$
and $N :: 'a$
and $R :: ('a, 'b, 'c) psi$
and $C :: 'f::fs-name$

assumes $Trans: \Psi \triangleright P \parallel Q \mapsto_i M(N) \prec R$

and $rPar1: \bigwedge P' A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P' \rrbracket; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M; A_Q \#* N; A_Q \#* C$

$\implies Prop (P' \parallel Q)$

and $rPar2: \bigwedge Q' A_P \Psi_P. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q' \rrbracket; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_P \#* N; A_P \#* C$

$\implies Prop (P \parallel Q')$

and $rBrMerge: \bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q.$

$\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P' \rrbracket; extractFrame P = \langle A_P, \Psi_P \rangle;$

distinct $A_P;$

$\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$

distinct $A_Q;$

$A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C \implies$
 $Prop (P' \parallel Q')$

shows $Prop R$

proof –

from $Trans$ **obtain** α **where** $\Psi \triangleright P \parallel Q \mapsto \alpha \prec R$ **and** $bn \alpha \#* \Psi$ **and** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ **and** $bn \alpha \#*$ *subject* α **and** $\alpha = M(N)$ **by** *auto*

then show *?thesis using rPar1 rPar2 rBrMerge*

by(*induct rule: parCases*) (*auto simp add: residualInject action.inject*)

qed

lemma *parOutputCases*[*consumes 5, case-names cPar1 cPar2*]:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $Q :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $xvec :: \text{name list}$
and $N :: 'a$
and $R :: ('a, 'b, 'c) \text{ psi}$
and $C :: 'f::\text{fs-name}$

assumes *Trans*: $\Psi \triangleright P \parallel Q \mapsto M(\nu*xvec)\langle N \rangle \prec R$

and $xvec \#* \Psi$
and $xvec \#* P$
and $xvec \#* Q$
and $xvec \#* M$

and $rPar1: \bigwedge P' A_Q \Psi_Q. [\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; \text{extractFrame}$
 $Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_Q;$

$A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M; A_Q \#* xvec; A_Q \#* N;$
 $A_Q \#* C; A_Q \#* xvec; \text{distinct } xvec] \implies \text{Prop } (P' \parallel Q)$

and $rPar2: \bigwedge Q' A_P \Psi_P. [\Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu*xvec)\langle N \rangle \prec Q'; \text{extractFrame}$
 $P = \langle A_P, \Psi_P \rangle; \text{distinct } A_P;$

$A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_P \#* xvec; A_P \#* N;$
 $A_P \#* C; A_P \#* xvec; \text{distinct } xvec] \implies \text{Prop } (P \parallel Q')$

shows *Prop R*

proof –

from *Trans* **have** *distinct xvec* **by**(*auto dest: boundOutputDistinct*)

obtain α **where** $\alpha = M(\nu*xvec)\langle N \rangle$ **by** *simp*

with *Trans* $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle xvec \#* M \rangle$

have $\Psi \triangleright P \parallel Q \mapsto \alpha \prec R$ **and** $bn \alpha \#* \Psi$ **and** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ **bn**
 $\alpha \#* \text{subject } \alpha$

by *simp+*

then show *?thesis* **using** $\langle \alpha = M(\nu*xvec)\langle N \rangle \rangle$ *rPar1 rPar2 distinct xvec*

by(*induct rule: parCases[where C=(xvec, C)]*) (*auto simp add: residualInject*)

qed

lemma *parBrOutputCases*[*consumes 5, case-names cPar1 cPar2 cBrComm1 cBrComm2*]:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $Q :: ('a, 'b, 'c) \text{ psi}$
and $M :: 'a$
and $xvec :: \text{name list}$
and $N :: 'a$
and $R :: ('a, 'b, 'c) \text{ psi}$
and $C :: 'f::\text{fs-name}$

assumes *Trans*: $\Psi \triangleright P \parallel Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec R$

and $xvec \#* \Psi$
and $xvec \#* P$

and $xvec \#* Q$
and $xvec \#* M$
and $rPar1: \bigwedge P' A_Q \Psi_Q. [\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'; extractFrame$
 $Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M; A_Q \#* xvec; A_Q \#* N;$
 $A_Q \#* C; A_Q \#* xvec; distinct xvec] \implies Prop (P' \parallel Q)$
and $rPar2: \bigwedge Q' A_P \Psi_P. [\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'; extractFrame$
 $P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_P \#* xvec; A_P \#* N;$
 $A_P \#* C; A_P \#* xvec; distinct xvec] \implies Prop (P \parallel Q')$
and $rBrComm1: \bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_i M\langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct$
 $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#*$
 $xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#*$
 $xvec; A_Q \#* Q'; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q] \implies$
 $Prop (P' \parallel Q')$
and $rBrComm2: \bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct$
 $A_Q;$
 $distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#*$
 $xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#*$
 $xvec; A_Q \#* Q'; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q] \implies$
 $Prop (P' \parallel Q')$

shows $Prop R$
proof –
from $Trans$ **have** $distinct xvec$ **by** $(auto dest: boundOutputDistinct)$
obtain α **where** $\alpha =_i M(\nu*xvec)\langle N \rangle$ **by** $simp$
with $Trans \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle xvec \#* M \rangle$
have $\Psi \triangleright P \parallel Q \mapsto_\alpha \prec R$ **and** $bn \alpha \#* \Psi$ **and** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ bn
 $\alpha \#*$ $subject \alpha$
by $simp+$
then show $?thesis$ **using** $\langle \alpha =_i M(\nu*xvec)\langle N \rangle \rangle rPar1 rPar2 rBrComm1 rBr-$
 $Comm2 \langle distinct xvec \rangle$
by $(induct rule: parCases[where C=(xvec, C)]) (auto simp add: residualInject$
 $action.inject)$
qed

```

lemma theEqvt[eqt-force]:
  fixes  $p :: \text{name prm}$ 
  and  $\alpha :: 'a \text{ action}$ 

assumes  $\alpha \neq \tau$ 

shows  $(p \cdot \text{the}(\text{subject } \alpha)) = \text{the}(p \cdot (\text{subject } \alpha))$ 
  using assms
  by(induct rule: actionCases[where  $\alpha=\alpha$ ]) auto

lemma theSubjectFresh[simp]:
  fixes  $\alpha :: 'a \text{ action}$ 
  and  $x :: \text{name}$ 

assumes  $\alpha \neq \tau$ 

shows  $x \# \text{the}(\text{subject } \alpha) = x \# \text{subject } \alpha$ 
  using assms
  by(cases rule: actionCases) auto

lemma theSubjectFreshChain[simp]:
  fixes  $\alpha :: 'a \text{ action}$ 
  and  $xvec :: \text{name list}$ 

assumes  $\alpha \neq \tau$ 

shows  $xvec \#* \text{the}(\text{subject } \alpha) = xvec \#* \text{subject } \alpha$ 
  using assms
  by(cases rule: actionCases) auto

lemma inputObtainPrefix:
  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $P' :: ('a, 'b, 'c) \text{ psi}$ 
  and  $A_P :: \text{name list}$ 
  and  $\Psi_P :: 'b$ 
  and  $N :: 'a$ 
  and  $K :: 'a$ 
  and  $B :: \text{name list}$ 

assumes  $\Psi \triangleright P \mapsto K(\downarrow N) \prec P'$ 
  and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
  and distinct  $A_P$ 
  and  $B \#* P$ 
  and  $A_P \#* \Psi$ 
  and  $A_P \#* B$ 
  and  $A_P \#* P$ 
  and  $A_P \#* K$ 

```

```

obtains  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  using assms
proof(nominal-induct avoiding: B arbitrary: thesis rule: inputFrameInduct)
  case(cAlpha  $\Psi P K N P' A_P \Psi_P p B$ )
  then obtain  $M$  where subjEq:  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by(auto intro: cAlpha)
  from  $\langle \Psi \otimes \Psi_P \vdash K \leftrightarrow M \rangle$ 
  have  $p \cdot (\Psi \otimes \Psi_P \vdash K \leftrightarrow M)$  by simp
  with  $\langle \text{set } p \subseteq \text{set } A_P \times \text{set } (p \cdot A_P) \rangle$ 
     $\langle A_P \#^* \Psi \rangle \langle (p \cdot A_P) \#^* \Psi \rangle \langle A_P \#^* K \rangle \langle (p \cdot A_P) \#^* K \rangle$ 
  have permEq:  $\Psi \otimes (p \cdot \Psi_P) \vdash K \leftrightarrow (p \cdot M)$  by(simp add: eqvts)
  from  $\langle B \#^* M \rangle$  have  $p \cdot (B \#^* M)$  by simp
  with  $\langle \text{set } p \subseteq \text{set } A_P \times \text{set } (p \cdot A_P) \rangle$ 
     $\langle A_P \#^* B \rangle \langle (p \cdot A_P) \#^* B \rangle$ 
  have permFresh:  $B \#^* (p \cdot M)$  by(simp add: eqvts)

  show ?case using cAlpha permEq permFresh
    by auto
next
  case(cInput  $\Psi M K xvec N Tvec P B$ )
  from  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$ 
    by(blast intro: statEqEnt AssertionStatEqSym[OF Identity])
  then have  $\Psi \otimes \mathbf{1} \vdash K \leftrightarrow M$  by(rule chanEqSym)
  moreover from  $\langle B \#^* (M(\lambda xvec N).P) \rangle$  have  $B \#^* M$  by simp
  ultimately show ?case by(auto intro: cInput)
next
  case(cCase  $\Psi P K N P' \varphi Cs A_P \Psi_P B$ )
  then obtain  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by – (rule cCase, auto dest: memFreshChain)
  with  $\langle \Psi_P \simeq \mathbf{1} \rangle$  show ?case by(blast intro: cCase statEqEnt compositionSym Identity)
next
  case(cPar1  $\Psi \Psi_Q P K N P' A_Q Q A_P \Psi_P B$ )
  then obtain  $M$  where  $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by (metis freshCompChain(1) psiFreshVec(4))
  then show ?case
    by(metis cPar1 statEqEnt Associativity Commutativity AssertionStatEqTrans Composition)
next
  case(cPar2  $\Psi \Psi_P Q K N Q' A_P P A_Q \Psi_Q B$ )
  then obtain  $M$  where  $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by – (rule cPar2, auto)
  then show ?case by(metis cPar2 statEqEnt Associativity)
next
  case(cScope  $\Psi P K N P' x A_P \Psi_P B$ )
  then obtain  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by – (rule cScope, auto)
  then show ?case by(auto intro: cScope)

```

```

next
  case(cBang  $\Psi$   $P$   $K$   $N$   $P'$   $A_P$   $\Psi_P$   $B$ )
  then obtain  $M$  where  $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by  $-$  (rule cBang, auto)
  with  $\langle \Psi_P \simeq \mathbf{1} \rangle$  show ?case by(metis cBang statEqEnt compositionSym Identity)
qed

```

lemma *outputObtainPrefix*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c)$  psi
  and  $P' :: ('a, 'b, 'c)$  psi
  and  $A_P ::$  name list
  and  $\Psi_P :: 'b$ 
  and  $N :: 'a$ 
  and  $K :: 'a$ 
  and  $xvec ::$  name list
  and  $B ::$  name list

```

```

assumes  $\Psi \triangleright P \mapsto ROut\ K\ ((\nu^*xvec)N \prec' P')$ 
  and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
  and distinct  $A_P$ 
  and  $xvec \#^* K$ 
  and distinct  $xvec$ 
  and  $B \#^* P$ 
  and  $A_P \#^* \Psi$ 
  and  $A_P \#^* B$ 
  and  $A_P \#^* P$ 
  and  $A_P \#^* K$ 

```

```

obtains  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  using assms

```

```

proof(nominal-induct avoiding: B xvec arbitrary: thesis rule: outputFrameInduct)
  case(cAlpha  $\Psi$   $P$   $K$   $A_P$   $\Psi_P$   $p$   $\alpha$   $B$   $xvec$ )
  then obtain  $M$  where subjEq:  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
    by(auto intro: cAlpha)

```

```

from  $\langle \Psi \otimes \Psi_P \vdash K \leftrightarrow M \rangle$ 
have  $p \cdot (\Psi \otimes \Psi_P \vdash K \leftrightarrow M)$  by simp

```

```

with  $\langle set\ p \subseteq set\ A_P \times set\ (p \cdot A_P) \rangle$ 
   $\langle A_P \#^* \Psi \rangle \langle (p \cdot A_P) \#^* \Psi \rangle \langle A_P \#^* K \rangle \langle (p \cdot A_P) \#^* K \rangle$ 
have permEq:  $\Psi \otimes (p \cdot \Psi_P) \vdash K \leftrightarrow (p \cdot M)$  by(simp add: eqts)

```

```

from  $\langle B \#^* M \rangle$  have  $p \cdot (B \#^* M)$  by simp
with  $\langle set\ p \subseteq set\ A_P \times set\ (p \cdot A_P) \rangle$ 
   $\langle A_P \#^* B \rangle \langle (p \cdot A_P) \#^* B \rangle$ 
have permFresh:  $B \#^* (p \cdot M)$  by(simp add: eqts)

```

```

show ?case using cAlpha permEq permFresh

```

```

    by auto
next
case(cOutput  $\Psi$   $M$   $K$   $N$   $P$   $B$   $xvec$ )
from  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \otimes \mathbf{1} \vdash M \leftrightarrow K$ 
  by(blast intro: statEqEnt AssertionStatEqSym[OF Identity])
then have  $\Psi \otimes \mathbf{1} \vdash K \leftrightarrow M$ 
  by(rule chanEqSym)
moreover from  $\langle B \#^* (M \langle N \rangle . P) \rangle$  have  $B \#^* M$  by simp
ultimately show ?case by(auto intro: cOutput)
next
case(cCase  $\Psi$   $P$   $K$   $P'$   $\varphi$   $Cs$   $A_P$   $\Psi_P$   $B$   $xvec$ )
then obtain  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  by - (rule cCase, auto dest: memFreshChain)
with  $\langle \Psi_P \simeq \mathbf{1} \rangle$  show ?case by(blast intro: cCase statEqEnt compositionSym
Identity)
next
case(cPar1  $\Psi$   $\Psi_Q$   $P$   $K$   $yvec$   $N$   $P'$   $A_Q$   $Q$   $A_P$   $\Psi_P$   $B$   $xvec$ )
then obtain  $M$  where  $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  by (metis freshCompChain(1) psiFreshVec(4))
then show ?case by(metis cPar1 statEqEnt Associativity Commutativity Asser-
tionStatEqTrans Composition)
next
case(cPar2  $\Psi$   $\Psi_P$   $Q$   $K$   $yvec$   $N$   $Q'$   $A_P$   $P$   $A_Q$   $\Psi_Q$   $B$   $xvec$ )
then obtain  $M$  where  $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  by (metis freshCompChain(1) psiFreshVec(4))
then show ?case by(metis cPar2 statEqEnt Associativity)
next
case(cOpen  $\Psi$   $P$   $M$   $zvec$   $yvec$   $N$   $P'$   $x$   $A_P$   $\Psi_P$   $B$   $xvec$ )
then obtain  $K$  where  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$  and  $B \#^* K$ 
  by (metis abs-fresh-list-star' psiFreshVec(5))
then show ?case by(auto intro: cOpen)
next
case(cScope  $\Psi$   $P$   $K$   $yvec$   $N$   $P'$   $x$   $A_P$   $\Psi_P$   $B$   $xvec$ )
then obtain  $M$  where  $\Psi \otimes \Psi_P \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  by (metis abs-fresh-list-star' psiFreshVec(5))
then show ?case by(auto intro: cScope)
next
case(cBang  $\Psi$   $P$   $K$   $P'$   $A_P$   $\Psi_P$   $B$   $xvec$ )
then obtain  $M$  where  $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash K \leftrightarrow M$  and  $B \#^* M$ 
  by - (rule cBang, auto)
with  $\langle \Psi_P \simeq \mathbf{1} \rangle$  show ?case by(metis cBang statEqEnt compositionSym Identity)
qed

```

lemma *inputRenameSubject*:

```

fixes  $\Psi$  :: 'b
  and  $P$  :: ('a, 'b, 'c) psi
  and  $M$  :: 'a
  and  $N$  :: 'a
  and  $P'$  :: ('a, 'b, 'c) psi

```

```

and  $A_P :: \text{name list}$ 
and  $\Psi_P :: 'b$ 

assumes  $\Psi \triangleright P \mapsto M(N) \prec P'$ 
and  $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ 
and  $\text{distinct } A_P$ 
and  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ 
and  $A_P \#^* \Psi$ 
and  $A_P \#^* P$ 
and  $A_P \#^* M$ 
and  $A_P \#^* K$ 

shows  $\Psi \triangleright P \mapsto K(N) \prec P'$ 
using  $\text{assms}$ 
proof( $\text{nominal-induct avoiding: } K \text{ rule: inputFrameInduct}$ )
case( $cAlpha \Psi P M N P' A_P \Psi_P p K$ )
have  $S: \text{set } p \subseteq \text{set } A_P \times \text{set } (p \cdot A_P)$  by  $\text{fact}$ 
from  $\langle \Psi \otimes (p \cdot \Psi_P) \vdash M \leftrightarrow K \rangle$  have  $(p \cdot (\Psi \otimes (p \cdot \Psi_P))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$ 
by( $\text{rule chanEqClosed}$ )
with  $S \langle \text{distinctPerm } p \rangle \langle A_P \#^* \Psi \rangle \langle A_P \#^* M \rangle \langle A_P \#^* K \rangle \langle (p \cdot A_P) \#^* \Psi \rangle \langle (p \cdot A_P) \#^* M \rangle \langle (p \cdot A_P) \#^* K \rangle$ 
have  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$  by( $\text{simp add: eqvts}$ )
with  $\langle A_P \#^* \Psi \rangle \langle A_P \#^* P \rangle \langle A_P \#^* M \rangle \langle A_P \#^* K \rangle$ 
 $\langle \llbracket \Psi \otimes \Psi_P \vdash M \leftrightarrow K; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* K \rrbracket \implies \Psi \triangleright P \mapsto K(N) \prec P' \rangle$ 
show  $?case$  by  $\text{blast}$ 
next
case( $cInput \Psi M K \text{vec } N \text{Tvec } P K'$ )
from  $\langle \Psi \otimes \mathbf{1} \vdash K \leftrightarrow K' \rangle$  have  $\Psi \vdash K \leftrightarrow K'$ 
by( $\text{blast intro: statEqEnt Identity}$ )
with  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \vdash M \leftrightarrow K'$ 
by( $\text{rule chanEqTrans}$ )
then show  $?case$  using  $\langle \text{distinct } \text{vec} \rangle \langle \text{set } \text{vec} \subseteq \text{supp } N \rangle \langle \text{length } \text{vec} = \text{length } \text{Tvec} \rangle$ 
by( $\text{rule Input}$ )
next
case( $cCase \Psi P M N P' \varphi Cs A_P \Psi_P K$ )
from  $\langle \Psi \otimes \mathbf{1} \vdash M \leftrightarrow K \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ 
by( $\text{blast intro: statEqEnt Identity compositionSym AssertionStatEqSym}$ )
with  $\langle A_P \#^* \Psi \rangle \langle A_P \#^* P \rangle \langle A_P \#^* M \rangle \langle A_P \#^* K \rangle$ 
 $\langle \bigwedge K. \llbracket \Psi \otimes \Psi_P \vdash M \leftrightarrow K; A_P \#^* \Psi; A_P \#^* P; A_P \#^* M; A_P \#^* K \rrbracket \implies \Psi \triangleright P \mapsto K(N) \prec P' \rangle$ 
have  $\Psi \triangleright P \mapsto K(N) \prec P'$  by  $\text{force}$ 
then show  $?case$  using  $\langle (\varphi, P) \in \text{set } Cs \rangle \langle \Psi \vdash \varphi \rangle \langle \text{guarded } P \rangle$  by( $\text{rule Case}$ )
next
case( $cPar1 \Psi \Psi_Q P M N P' A_Q Q A_P \Psi_P K$ )
from  $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$  have  $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K$ 
by( $\text{metis statEqEnt Associativity Composition AssertionStatEqTrans Commu-}$ 

```

tativity)

```

with  $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle$ 
   $\langle \bigwedge K. [(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K; A_P \#* (\Psi \otimes \Psi_Q); A_P \#* P; A_P \#* M; A_P \#* K] \implies \Psi \otimes \Psi_Q \triangleright P \mapsto K(N) \prec P' \rangle$ 
have  $\Psi \otimes \Psi_Q \triangleright P \mapsto K(N) \prec P'$  by force
then show  $?case$  using  $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle$ 
 $\langle A_Q \#* K \rangle \langle A_Q \#* N \rangle$ 
by(auto intro: Par1)
next
case(cPar2  $\Psi \Psi_P Q M N Q' A_P P A_Q \Psi_Q K$ )
from  $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$  have  $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K$ 
by(rule statEqEnt[OF AssertionStatEqSym[OF Associativity]])
with  $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle A_Q \#* K \rangle$ 
   $\langle \bigwedge K. [(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K; A_Q \#* (\Psi \otimes \Psi_P); A_Q \#* Q; A_Q \#* M; A_Q \#* K] \implies \Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q' \rangle$ 
have  $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'$  by force
then show  $?case$  using  $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* Q \rangle$ 
 $\langle A_P \#* K \rangle \langle A_P \#* N \rangle$ 
by(auto intro: Par2)
next
case(cScope  $\Psi P M N P' x A_P \Psi_P$ )
then have  $\Psi \triangleright P \mapsto K(N) \prec P'$  by force
with  $\langle x \# \Psi \rangle \langle x \# K \rangle \langle x \# N \rangle$  show  $?case$ 
by(auto intro: Scope)
next
case(cBang  $\Psi P M N P' A_P \Psi_P K$ )
from  $\langle \Psi \otimes \mathbf{1} \vdash M \leftrightarrow K \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K$ 
by(blast intro: statEqEnt Identity compositionSym AssertionStatEqSym)
with  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle$ 
   $\langle \bigwedge K. [\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* (P \parallel !P); A_P \#* M; A_P \#* K] \implies \Psi \triangleright P \parallel !P \mapsto K(N) \prec P' \rangle$ 
have  $\Psi \triangleright P \parallel !P \mapsto K(N) \prec P'$  by force
then show  $?case$  using  $\langle guarded P \rangle$  by(rule Bang)
qed

```

lemma *outputRenameSubject:*

```

fixes  $\Psi$   $:: 'b$ 
and  $P$   $:: ('a, 'b, 'c) psi$ 
and  $M$   $:: 'a$ 
and  $xvec$   $:: name list$ 
and  $N$   $:: 'a$ 
and  $P'$   $:: ('a, 'b, 'c) psi$ 
and  $A_P$   $:: name list$ 
and  $\Psi_P$   $:: 'b$ 

```

```

assumes  $\Psi \triangleright P \mapsto M(\nu*xvec)(N) \prec P'$ 
and  $extractFrame P = \langle A_P, \Psi_P \rangle$ 
and  $distinct A_P$ 
and  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ 

```

```

and  $A_P \#* \Psi$ 
and  $A_P \#* P$ 
and  $A_P \#* M$ 
and  $A_P \#* K$ 

shows  $\Psi \triangleright P \mapsto K(\nu*xvec)\langle N \rangle \prec P'$ 
  using assms unfolding residualInject
proof(nominal-induct avoiding: K rule: outputFrameInduct)
  case(cAlpha  $\Psi P M A_P \Psi_P p B K$ )
  have  $S: \text{set } p \subseteq \text{set } A_P \times \text{set}(p \cdot A_P)$  by fact
  from  $\langle \Psi \otimes (p \cdot \Psi_P) \vdash M \leftrightarrow K \rangle$  have  $(p \cdot (\Psi \otimes (p \cdot \Psi_P))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$ 
    by(rule chanEqClosed)
  with  $S \langle \text{distinctPerm } p \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle \langle (p \cdot A_P) \#* \Psi \rangle \langle (p \cdot A_P) \#* M \rangle \langle (p \cdot A_P) \#* K \rangle$ 
  have  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$  by(simp add: eqvts)
  with  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle$ 
  show ?case by(blast intro: cAlpha)
next
  case(cOutput  $\Psi M K N P K'$ )
  from  $\langle \Psi \otimes \mathbf{1} \vdash K \leftrightarrow K' \rangle$  have  $\Psi \vdash K \leftrightarrow K'$ 
    by(blast intro: statEqEnt Identity)
  with  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \vdash M \leftrightarrow K'$ 
    by(rule chanEqTrans)
  then show ?case using Output by(force simp add: residualInject)
next
  case(cCase  $\Psi P M B \varphi Cs A_P \Psi_P K$ )
  from  $\langle \Psi \otimes \mathbf{1} \vdash M \leftrightarrow K \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$ 
    by(blast intro: statEqEnt Identity compositionSym AssertionStatEqSym)
  with  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle$ 
   $\langle \bigwedge K. [\Psi \otimes \Psi_P \vdash M \leftrightarrow K; A_P \#* \Psi; A_P \#* P; A_P \#* M; A_P \#* K] \implies \Psi \triangleright P \mapsto (R\text{Out } K B) \rangle$ 
  have  $\Psi \triangleright P \mapsto R\text{Out } K B$  by force
  then show ?case using  $\langle (\varphi, P) \in \text{set } Cs \rangle \langle \Psi \vdash \varphi \rangle \langle \text{guarded } P \rangle$  by(rule Case)
next
  case(cPar1  $\Psi \Psi_Q P M xvec N P' A_Q Q A_P \Psi_P K$ )
  from  $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$  have  $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K$ 
    by(metis statEqEnt Associativity Composition AssertionStatEqTrans Commutativity)
  with  $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K \rangle$ 
   $\langle \bigwedge K. [(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K; A_P \#* (\Psi \otimes \Psi_Q); A_P \#* P; A_P \#* M; A_P \#* K] \implies \Psi \otimes \Psi_Q \triangleright P \mapsto (R\text{Out } K (\nu*xvec)\langle N \rangle \prec' P') \rangle$ 
  have  $\Psi \otimes \Psi_Q \triangleright P \mapsto K(\nu*xvec)\langle N \rangle \prec P'$  by(force simp add: residualInject)
  then show ?case using  $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle xvec \#* Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* K \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* N \rangle$  Par1 where  $\alpha = K(\nu*xvec)\langle N \rangle$ 
    by(auto simp add: residualInject)
next
  case(cPar2  $\Psi \Psi_P Q M xvec N Q' A_P P A_Q \Psi_Q K$ )
  from  $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$  have  $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash M \leftrightarrow K$ 

```



```

  by(rule statEqEnt[OF AssertionStatEqSym[OF Associativity]])
  with ⟨AQ #* Ψ⟩ ⟨AQ #* ΨP⟩ ⟨AQ #* Q⟩ ⟨AQ #* M⟩ ⟨AQ #* K⟩
    ⟨∧K. [(Ψ ⊗ ΨP) ⊗ ΨQ ⊢ M ↔ K; AQ #* (Ψ ⊗ ΨP); AQ #* Q; AQ #* M; AQ
#* K] ⇒ Ψ ⊗ ΨP ▷ Q ⟶ ROut K ((ν*xvec)N <' Q')⟩
  have Ψ ⊗ ΨP ▷ Q ⟶ ROut K ((ν*xvec)N <' Q') by force
  then show ?case using ⟨extractFrame P = ⟨AP, ΨP⟩⟩ ⟨xvec #* P⟩ ⟨AP #* Ψ⟩
    ⟨AP #* Q⟩ ⟨AP #* K⟩ ⟨AP #* xvec⟩ ⟨AP #* N⟩ Par2[where α=K((ν*xvec)⟨N⟩)]
    by(auto simp add: residualInject)
next
case(cOpen Ψ P M xvec yvec N P' x AP ΨP)
  then have Ψ ▷ P ⟶ K((ν*(xvec@yvec))⟨N⟩) < P' by(force simp add: residual-
Inject)
  with ⟨x ∈ supp N⟩ ⟨x # Ψ⟩ ⟨x # K⟩ ⟨x # xvec⟩ ⟨x # yvec⟩ Open show ?case
    by(auto simp add: residualInject)
next
case(cScope Ψ P M xvec N P' x AP ΨP)
  then have Ψ ▷ P ⟶ K((ν*xvec)⟨N⟩) < P' by(force simp add: residualInject)
  with ⟨x # Ψ⟩ ⟨x # K⟩ ⟨x # xvec⟩ ⟨x # N⟩ Scope[where α=K((ν*xvec)⟨N⟩)] show
?case
  by(auto simp add: residualInject)
next
case(cBang Ψ P M B AP ΨP K)
  from ⟨Ψ ⊗ 1 ⊢ M ↔ K⟩ ⟨ΨP ≃ 1⟩ have Ψ ⊗ ΨP ⊗ 1 ⊢ M ↔ K
  by(blast intro: statEqEnt Identity compositionSym AssertionStatEqSym)
  with ⟨AP #* Ψ⟩ ⟨AP #* P⟩ ⟨AP #* M⟩ ⟨AP #* K⟩
    ⟨∧K. [Ψ ⊗ ΨP ⊗ 1 ⊢ M ↔ K; AP #* Ψ; AP #* (P || !P); AP #* M; AP #*
K] ⇒ Ψ ▷ P || !P ⟶ ROut K B⟩
  have Ψ ▷ P || !P ⟶ ROut K B by force
  then show ?case using ⟨guarded P⟩ by(rule Bang)
qed

```

lemma *parCasesSubject*[consumes 7, case-names *cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2*]:

```

  fixes Ψ      :: 'b
  and P        :: ('a, 'b, 'c) psi
  and Q        :: ('a, 'b, 'c) psi
  and α        :: 'a action
  and R        :: ('a, 'b, 'c) psi
  and C        :: 'f::fs-name
  and yvec     :: name list

```

```

assumes Trans: Ψ ▷ P || Q ⟶ α < R
and      bn α #* Ψ
and      bn α #* P
and      bn α #* Q
and      bn α #* subject α
and      yvec #* P
and      yvec #* Q
and      rPar1: ∧P' AQ ΨQ. [Ψ ⊗ ΨQ ▷ P ⟶ α < P'; extractFrame Q = ⟨AQ,

```

Ψ_Q); *distinct* A_Q ;
 $A_Q \#* \Psi; A_Q \#* P; A_Q \#* \alpha; A_Q \#* C \implies Prop \alpha (P' \parallel Q)$
and *rPar2*: $\bigwedge Q' A_P \Psi_P. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $A_P \#* \Psi; A_P \#* Q; A_P \#* \alpha; A_P \#* C \implies Prop \alpha (P \parallel Q')$
and *rComm1*: $\bigwedge \Psi_Q M N P' A_P \Psi_P K xvec Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct A_Q ;
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; yvec \#* M; yvec \#* K; distinct xvec; \alpha = \tau;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* Q;$
 $A_P \#* xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q;$
 $A_Q \#* xvec; A_Q \#* Q'; A_Q \#* C;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* Q;$
 $xvec \#* \Psi_Q; xvec \#* C \implies$
 $Prop (\tau) (\nu * xvec)(P' \parallel Q')$
and *rComm2*: $\bigwedge \Psi_Q M xvec N P' A_P \Psi_P K Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
distinct A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_Q;$
 $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; yvec \#* M; yvec \#* K; distinct xvec; \alpha = \tau;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* M; A_P \#* N; A_P \#* P'; A_P \#* Q;$
 $A_P \#* xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* K; A_Q \#* N; A_Q \#* P'; A_Q \#* Q;$
 $A_Q \#* xvec; A_Q \#* Q'; A_Q \#* C;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* Q;$
 $xvec \#* \Psi_Q; xvec \#* C \implies$
 $Prop (\tau) (\nu * xvec)(P' \parallel Q')$
and *rBrMerge*: $\bigwedge \Psi_Q M N P' A_P \Psi_P Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
distinct A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct A_Q ;
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P';$
 $A_P \#* Q; A_P \#* Q'; A_P \#* A_Q; A_P \#* M; A_Q \#* M;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P';$
 $A_Q \#* Q; A_Q \#* Q'; A_P \#* C; A_Q \#* C; \alpha = \iota M(N) \implies$
 $Prop (\iota M(N)) (P' \parallel Q')$
and *rBrComm1*: $\bigwedge \Psi_Q M N P' A_P \Psi_P xvec Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct$
 A_P ;
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\nu * xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
distinct A_Q ;
distinct $xvec$;
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#* xvec;$
 $A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#* xvec;$
 $A_Q \#* Q'; A_Q \#* C;$

$A_P \#* M; A_Q \#* M; xvec \#* M; \text{j}M(\nu*xvec)\langle N \rangle = \alpha;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q \implies$
 $\text{Prop}(\text{j}M(\nu*xvec)\langle N \rangle) (P' \parallel Q')$
and $rBrComm2: \wedge \Psi_Q M xvec N P' A_P \Psi_P Q' A_Q.$
 $\llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \text{j}M(\nu*xvec)\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle;$
distinct $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto \text{j}M(N) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$ *distinct*
 $A_Q;$
distinct $xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* N; A_P \#* P'; A_P \#* Q; A_P \#*$
 $xvec; A_P \#* Q'; A_P \#* A_Q; A_P \#* C;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* N; A_Q \#* P'; A_Q \#* Q; A_Q \#*$
 $xvec; A_Q \#* Q'; A_Q \#* C;$
 $A_P \#* M; A_Q \#* M; xvec \#* M; \text{j}M(\nu*xvec)\langle N \rangle = \alpha;$
 $xvec \#* \Psi; xvec \#* \Psi_P; xvec \#* P; xvec \#* Q; xvec \#* \Psi_Q \implies$
 $\text{Prop}(\text{j}M(\nu*xvec)\langle N \rangle) (P' \parallel Q')$

shows $\text{Prop } \alpha R$
using $\text{Trans} \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* \text{subject } \alpha \rangle$
proof(*induct rule: parCases*[**where** $C=(C, yvec)$])
case($cPar1 P' A_Q \Psi_Q$)
then show $?case$ **by**(*auto intro: rPar1*)
next
case($cPar2 Q' A_P \Psi_P$)
then show $?case$ **by**(*auto intro: rPar2*)
next
case($cComm1 \Psi_Q M N P' A_P \Psi_P K xvec Q' A_Q$)
from $\langle A_P \#* (C, yvec) \rangle \langle A_Q \#* (C, yvec) \rangle \langle xvec \#* (C, yvec) \rangle$
have $A_P \#* C$ **and** $A_Q \#* C$ **and** $xvec \#* C$ **and** $A_P \#* yvec$ **and** $A_Q \#* yvec$
and $xvec \#* yvec$
by $\text{simp}+$

have $FrP: \text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** $FrQ: \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$
and $MeqK: \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ **by** $\text{fact}+$

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \text{j}M(N) \prec P' \rangle FrP \langle \text{distinct } A_P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$
 $\langle yvec \#* P \rangle \langle A_P \#* \Psi \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_P \#* yvec \rangle \langle A_P \#* xvec \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle xvec \#* P \rangle \langle A_P$
 $\#* \Psi_Q \rangle$
obtain M' **where** $MeqM': (\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $xvec \#* M'$ **and**
 $yvec \#* M'$ **and** $A_Q \#* M'$
by $-$ (*rule inputObtainPrefix*[**where** $B=xvec@yvec@A_Q$], (*assumption | force*)+)
from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{j}M(\nu*xvec)\langle N \rangle \prec Q' \rangle$ **have** $\Psi \otimes \Psi_P \triangleright Q \mapsto ROut K$
 $(\text{j}M(\nu*xvec)\langle N \rangle \prec' Q')$
by(*simp add: residualInject*)
with $FrQ \langle \text{distinct } A_Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle yvec \#* Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_Q \#* yvec \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle xvec \#* Q \rangle \langle A_Q$
 $\#* \Psi_P \rangle \langle xvec \#* K \rangle \langle \text{distinct } xvec \rangle$
obtain K' **where** $KeqK': (\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $xvec \#* K'$ **and** $yvec$

$\#* K'$ and $A_P \#* K'$
by – (rule *outputObtainPrefix*[**where** $B=xvec@yvec@A_P$], (*assumption* | *force* | *metis freshChainSym*)+)

from $MeqK \ KeqK'$ **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
with $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu N) \prec P' \rangle \ FrP \ \langle distinct \ A_P \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto K'(\nu N) \prec P'$ **using** $\langle A_P \#* \Psi \rangle \ \langle A_P \#* \Psi_Q \rangle \ \langle A_P \#* P \rangle$
 $\langle A_P \#* M \rangle \ \langle A_P \#* K' \rangle$
by – (rule *inputRenameSubject*, (*assumption* | *force*)+)
moreover note $FrP \ \langle distinct \ A_P \rangle$
moreover from $MeqK \ MeqM'$ **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
with $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu xvec) \langle N \rangle \prec Q' \rangle \ FrQ \ \langle distinct \ A_Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto M'(\nu xvec) \langle N \rangle \prec Q'$ **using** $\langle A_Q \#* \Psi \rangle \ \langle A_Q \#* \Psi_P \rangle \ \langle A_Q \#* Q \rangle \ \langle A_Q \#* K \rangle \ \langle A_Q \#* M' \rangle$
by – (rule *outputRenameSubject*, (*assumption* | *force*)+)
moreover note $FrQ \ \langle distinct \ A_Q \rangle$
moreover from $MeqM' \ KeqK' \ MeqK$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
moreover note $\langle A_P \#* \Psi \rangle \ \langle A_P \#* \Psi_Q \rangle \ \langle A_P \#* P \rangle \ \langle A_P \#* K' \rangle \ \langle A_P \#* N \rangle \ \langle A_P \#* P' \rangle \ \langle A_P \#* Q \rangle \ \langle A_P \#* xvec \rangle \ \langle A_P \#* Q' \rangle \ \langle A_P \#* A_Q \rangle \ \langle A_P \#* C \rangle$
 $\langle A_Q \#* \Psi \rangle \ \langle A_Q \#* \Psi_P \rangle \ \langle A_Q \#* Q \rangle \ \langle A_Q \#* M' \rangle \ \langle A_Q \#* N \rangle \ \langle A_Q \#* Q' \rangle \ \langle A_Q \#* P \rangle \ \langle A_Q \#* xvec \rangle \ \langle A_Q \#* P' \rangle \ \langle A_Q \#* C \rangle \ \langle \alpha = \tau \rangle$
 $\langle xvec \#* \Psi \rangle \ \langle xvec \#* \Psi_P \rangle \ \langle xvec \#* P \rangle \ \langle xvec \#* M' \rangle \ \langle xvec \#* K' \rangle \ \langle xvec \#* Q \rangle$
 $\langle xvec \#* \Psi_Q \rangle \ \langle xvec \#* C \rangle \ \langle yvec \#* M' \rangle \ \langle yvec \#* K' \rangle \ \langle distinct \ xvec \rangle$
ultimately show *?case*
by(*metis rComm1*)

next
case(*cComm2* $\Psi_Q \ M \ xvec \ N \ P' \ A_P \ \Psi_P \ K \ Q' \ A_Q$)
from $\langle A_P \#* (C, yvec) \rangle \ \langle A_Q \#* (C, yvec) \rangle \ \langle xvec \#* (C, yvec) \rangle$
have $A_P \#* C$ and $A_Q \#* C$ and $xvec \#* C$ and $A_P \#* yvec$ and $A_Q \#* yvec$
and $xvec \#* yvec$
by *simp*+

have FrP : *extractFrame* $P = \langle A_P, \Psi_P \rangle$ and FrQ : *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
and $MeqK$: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$ **by** *fact*+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu xvec) \langle N \rangle \prec P' \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright P \mapsto ROut \ M$
 $(\nu xvec) \langle N \rangle \prec P'$
by(*simp add: residualInject*)
with $FrP \ \langle distinct \ A_P \rangle \ \langle A_P \#* P \rangle \ \langle A_Q \#* P \rangle \ \langle yvec \#* P \rangle \ \langle A_P \#* \Psi \rangle$
 $\langle A_P \#* A_Q \rangle \ \langle A_P \#* yvec \rangle \ \langle A_P \#* xvec \rangle \ \langle A_P \#* P \rangle \ \langle A_P \#* M \rangle \ \langle xvec \#* P \rangle \ \langle A_P \#* \Psi_Q \rangle \ \langle xvec \#* M \rangle \ \langle distinct \ xvec \rangle$
obtain M' **where** $MeqM'$: $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ and $xvec \#* M'$ and $yvec \#* M'$ and $A_Q \#* M'$
by – (rule *outputObtainPrefix*[**where** $B=xvec@yvec@A_Q$], (*assumption* | *force*)+)

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q' \rangle \text{FrQ} \langle \text{distinct } A_Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$
 $\langle \text{yvec} \#* Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_P \#* A_Q \rangle \langle A_Q \#* \text{yvec} \rangle \langle A_Q \#* \text{xvec} \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle \text{xvec} \#* Q \rangle \langle A_Q$
 $\#* \Psi_P \rangle$
obtain K' **where** $\text{KeqK}' : (\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $\text{xvec} \#* K'$ **and** yvec
 $\#* K'$ **and** $A_P \#* K'$
by $-$ (rule *inputObtainPrefix*[**where** $B = \text{xvec} @ \text{yvec} @ A_P$], (*assumption* | *force* |
metis freshChainSym)+)

from $\text{MeqK} \text{KeqK}'$ **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
with $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * \text{xvec})(N) \prec P' \rangle \text{FrP} \langle \text{distinct } A_P \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto K'(\nu * \text{xvec})(N) \prec P'$ **using** $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P$
 $\#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K' \rangle$
by $-$ (rule *outputRenameSubject*, (*assumption* | *force*)+)
moreover note $\text{FrP} \langle \text{distinct } A_P \rangle$
moreover from $\text{MeqK} \text{MeqM}'$ **have** $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*
chanEqSym)
with $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q' \rangle \text{FrQ} \langle \text{distinct } A_Q \rangle$
have $\Psi \otimes \Psi_P \triangleright Q \mapsto M'(N) \prec Q'$ **using** $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle$
 $\langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$
by $-$ (rule *inputRenameSubject*, (*assumption* | *force*)+)
moreover note $\text{FrQ} \langle \text{distinct } A_Q \rangle$
moreover from $\text{MeqM}' \text{KeqK}' \text{MeqK}$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition chanEqTrans*
chanEqSym)
moreover note $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* K' \rangle \langle A_P \#* N \rangle \langle A_P$
 $\#* P' \rangle \langle A_P \#* Q \rangle \langle A_P \#* \text{xvec} \rangle \langle A_P \#* Q' \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* C \rangle$
 $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M' \rangle \langle A_Q \#* N \rangle \langle A_Q \#* Q' \rangle \langle A_Q \#*$
 $P \rangle \langle A_Q \#* \text{xvec} \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* C \rangle \langle \alpha = \tau \rangle$
 $\langle \text{xvec} \#* \Psi \rangle \langle \text{xvec} \#* \Psi_P \rangle \langle \text{xvec} \#* P \rangle \langle \text{xvec} \#* M' \rangle \langle \text{xvec} \#* K' \rangle \langle \text{xvec} \#* Q \rangle$
 $\langle \text{xvec} \#* \Psi_Q \rangle \langle \text{xvec} \#* C \rangle \langle \text{yvec} \#* M' \rangle \langle \text{yvec} \#* K' \rangle \langle \text{distinct } \text{xvec} \rangle$
ultimately show *?case*
by(*metis rComm2*)

next
case *cBrMerge*
then show *?case* **by** (*simp add: rBrMerge*)
next
case *cBrComm1*
then show *?case* **by** (*auto intro: rBrComm1*)
next
case *cBrComm2*
then show *?case* **by** (*auto intro: rBrComm2*)
qed

lemma *inputCases*[*consumes 1*, *case-names cInput cBrInput*]:
fixes $\Psi \quad :: 'b$
and $M \quad :: 'a$

```

and xvec :: name list
and N    :: 'a'
and P    :: ('a', 'b', 'c') psi
and  $\alpha$   :: 'a' action
and P'   :: ('a', 'b', 'c') psi

assumes Trans:  $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto \alpha \prec P'$ 
and rInput:  $\bigwedge K\ Tvec. \llbracket \Psi \vdash M \leftrightarrow K; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec \rrbracket \implies Prop\ (K(N[xvec::=Tvec]))\ (P[xvec::=Tvec])$ 
and rBrInput:  $\bigwedge K\ Tvec. \llbracket \Psi \vdash K \succeq M; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec \rrbracket \implies Prop\ (\iota K(N[xvec::=Tvec]))\ (P[xvec::=Tvec])$ 

shows Prop  $\alpha\ P'$ 
proof -
  {
    fix xvec N P
    assume Trans:  $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto \alpha \prec P'$ 
    and xvec  $\#* \Psi$  and xvec  $\#* M$  and xvec  $\#* \alpha$  and xvec  $\#* P'$  and distinct
xvec
    and rInput:  $\bigwedge K\ Tvec. \llbracket \Psi \vdash M \leftrightarrow K; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec \rrbracket \implies Prop\ (K(N[xvec::=Tvec]))\ (P[xvec::=Tvec])$ 
    and rBrInput:  $\bigwedge K\ Tvec. \llbracket \Psi \vdash K \succeq M; set\ xvec \subseteq supp\ N; length\ xvec = length\ Tvec; distinct\ xvec \rrbracket \implies Prop\ (\iota K(N[xvec::=Tvec]))\ (P[xvec::=Tvec])$ 

    from Trans have bn  $\alpha = []$ 
    apply -
    by(ind-cases  $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto \alpha \prec P'$ ) (auto simp add: residualInject)
    from Trans have distinct(bn  $\alpha$ ) by(auto dest: boundOutputDistinct)
    have length(bn  $\alpha$ ) = residualLength( $\alpha \prec P'$ ) by simp
    note Trans
    moreover have length xvec = inputLength( $M(\lambda * xvec\ N).P$ ) by auto
    moreover note  $\langle distinct\ xvec \rangle$ 
    moreover have length xvec = inputLength( $M(\lambda * xvec\ N).P$ ) by auto
    moreover note  $\langle distinct\ xvec \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover note  $\langle length(bn\ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn\ \alpha) \rangle$ 
    moreover obtain x::name where x  $\# \Psi$  and x  $\# P$  and x  $\# M$  and x  $\# xvec$ 
and x  $\# \alpha$  and x  $\# P'$  and x  $\# N$ 
    by(generate-fresh name) auto
    ultimately have Prop  $\alpha\ P'$  using  $\langle bn\ \alpha = [] \rangle \langle xvec\ \#* \Psi \rangle \langle xvec\ \#* M \rangle \langle xvec\ \#* \alpha \rangle \langle xvec\ \#* P' \rangle$ 
    apply(cases rule: semanticsCases[of - - - - - C x x x x])
    apply(force simp add: residualInject psi.inject rInput)
    apply(force simp add: residualInject psi.inject rBrInput)
  }

```

```

    by(auto simp add: residualInject psi.inject inputChainFresh)+
  }
  note Goal = this
  moreover obtain p :: name prm where (p · xvec) #* Ψ and (p · xvec) #* M
and (p · xvec) #* N and (p · xvec) #* P
  and (p · xvec) #* α and (p · xvec) #* P' and S: set p ⊆ set xvec × set(p ·
xvec)
  and distinctPerm p
  by(rule name-list-avoiding[where xvec=xvec and c=(Ψ, M, N, P, α, P')])
auto
  from Trans ⟨(p · xvec) #* N⟩ ⟨(p · xvec) #* P⟩ S have Ψ ▷ M(λ*(p · xvec) (p
· N)).(p · P) ↦α < P'
  by(simp add: inputChainAlpha')
  moreover {
    fix K Tvec
    assume Ψ ⊢ M ↔ K
    moreover assume set(p · xvec) ⊆ supp(p · N)
    then have (p · set(p · xvec)) ⊆ (p · supp(p · N)) by simp
    with ⟨distinctPerm p⟩ have set xvec ⊆ supp N by(simp add: eqvts)
    moreover assume length(p · xvec) = length(Tvec::'a list)
    then have length xvec = length Tvec by simp
    moreover assume distinct xvec
    ultimately have Prop (K(N[xvec::=Tvec])) (P[xvec::=Tvec])
      by(rule rInput)
    with ⟨length xvec = length Tvec⟩ S ⟨distinctPerm p⟩ ⟨(p · xvec) #* N⟩ ⟨(p ·
xvec) #* P⟩
    have Prop (K((p · N)[(p · xvec)::=Tvec])) ((p · P)[(p · xvec)::=Tvec])
      by(simp add: renaming substTerm.renaming)
  }
  moreover {
    fix K Tvec
    assume Ψ ⊢ K ≽ M
    moreover assume set(p · xvec) ⊆ supp(p · N)
    then have (p · set(p · xvec)) ⊆ (p · supp(p · N)) by simp
    with ⟨distinctPerm p⟩ have set xvec ⊆ supp N by(simp add: eqvts)
    moreover assume length(p · xvec) = length(Tvec::'a list)
    then have length xvec = length Tvec by simp
    moreover assume distinct xvec
    ultimately have Prop (K(N[xvec::=Tvec])) (P[xvec::=Tvec])
      by(rule rBrInput)
    with ⟨length xvec = length Tvec⟩ S ⟨distinctPerm p⟩ ⟨(p · xvec) #* N⟩ ⟨(p ·
xvec) #* P⟩
    have Prop (K((p · N)[(p · xvec)::=Tvec])) ((p · P)[(p · xvec)::=Tvec])
      by(simp add: renaming substTerm.renaming)
  }
  moreover from Trans have distinct xvec by(rule inputDistinct)
  then have distinct(p · xvec) by simp
  ultimately show ?thesis using ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec) #* M⟩ ⟨(p · xvec)
#* α⟩ ⟨(p · xvec) #* P'⟩ ⟨distinct xvec⟩

```

by(*metis Goal*)
qed

lemma *outputCases*[*consumes 1, case-names cOutput cBrOutput*]:

fixes $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $\alpha :: 'a \text{ action}$
and $P' :: ('a, 'b, 'c) \text{ psi}$

assumes $\Psi \triangleright M \langle N \rangle . P \mapsto \alpha \prec P'$
and $\bigwedge K. \Psi \vdash M \leftrightarrow K \implies \text{Prop } (K \langle N \rangle) P$
and $\bigwedge K. \Psi \vdash M \preceq K \implies \text{Prop } (\text{i}K \langle N \rangle) P$

shows $\text{Prop } \alpha P'$
using *assms*
by(*cases rule: semantics.cases*) (*auto simp add: residualInject psi.inject*)

lemma *caseCases*[*consumes 1, case-names cCase*]:

fixes $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c) \text{ psi}) \text{ list}$
and $\alpha :: 'a \text{ action}$
and $P' :: ('a, 'b, 'c) \text{ psi}$

assumes *Trans*: $\Psi \triangleright (\text{Cases } Cs) \mapsto Rs$
and *rCase*: $\bigwedge \varphi P. \llbracket \Psi \triangleright P \mapsto Rs; (\varphi, P) \in \text{set } Cs; \Psi \vdash \varphi; \text{guarded } P \rrbracket \implies \text{Prop}$

shows *Prop*
using *assms*
by(*cases rule: semantics.cases*) (*auto simp add: residualInject psi.inject*)

lemma *resCases*[*consumes 7, case-names cOpen cBrOpen cRes cBrClose*]:

fixes $\Psi :: 'b$
and $x :: \text{name}$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $\alpha :: 'a \text{ action}$
and $P' :: ('a, 'b, 'c) \text{ psi}$
and $C :: 'f :: \text{fs-name}$

assumes *Trans*: $\Psi \triangleright (\nu x) P \mapsto \alpha \prec P'$
and $x \# \Psi$
and $x \# \alpha$
and $x \# P'$
and $\text{bn } \alpha \#* \Psi$
and $\text{bn } \alpha \#* P$
and $\text{bn } \alpha \#* \text{subject } \alpha$
and *rOpen*: $\bigwedge M \text{ xvec } \text{yvec } y N P'. \llbracket \Psi \triangleright P \mapsto M(\nu*(\text{xvec}@\text{yvec})) \langle [(x, y)] \rangle \cdot$

$N\rangle \prec ((x, y) \cdot P')$; $y \in \text{supp } N$;
 $x \# N$; $x \# P'$; $x \neq y$; $y \# \text{xvec}$; $y \# \text{yvec}$; $y \# M$;
distinct xvec; distinct yvec;
 $\text{xvec} \#* \Psi$; $y \# \Psi$; $\text{yvec} \#* \Psi$; $\text{xvec} \#* P$; $y \# P$;
 $\text{yvec} \#* P$; $\text{xvec} \#* M$; $y \# M$;
 $\text{yvec} \#* M$; $\text{xvec} \#* \text{yvec}$ \implies
 $\text{Prop } (M(\nu*(\text{xvec}@y\#\text{yvec}))\langle N \rangle) P'$
and *rBrOpen*: $\bigwedge M \text{xvec yvec } y N P'$. $\llbracket \Psi \triangleright P \mapsto \text{i}M(\nu*(\text{xvec}@y\#\text{yvec}))\langle (x, y) \cdot N \rangle \prec ((x, y) \cdot P') \text{; } y \in \text{supp } N$;
 $x \# N$; $x \# P'$; $x \neq y$; $y \# \text{xvec}$; $y \# \text{yvec}$; $y \# M$;
distinct xvec; distinct yvec;
 $\text{xvec} \#* \Psi$; $y \# \Psi$; $\text{yvec} \#* \Psi$; $\text{xvec} \#* P$; $y \# P$;
 $\text{yvec} \#* P$; $\text{xvec} \#* M$; $y \# M$;
 $\text{yvec} \#* M$; $\text{xvec} \#* \text{yvec}$ \implies
 $\text{Prop } (\text{i}M(\nu*(\text{xvec}@y\#\text{yvec}))\langle N \rangle) P'$
and *rScope*: $\bigwedge P'$. $\llbracket \Psi \triangleright P \mapsto \alpha \prec P \rrbracket \implies \text{Prop } \alpha ((\nu x)P')$
and *rBrClose*: $\bigwedge M \text{xvec } N P'$.
 $\llbracket \Psi \triangleright P \mapsto \text{i}M(\nu*\text{xvec})\langle N \rangle \prec P'$;
 $x \in \text{supp } M$;
distinct xvec; xvec # Psi; xvec #* P*;
 $\text{xvec} \#* M$;
 $x \# \Psi$; $x \# \text{xvec}$;
 $\text{xvec} \#* C \rrbracket \implies \text{Prop } (\tau) ((\nu x)((\nu*\text{xvec})P'))$

shows $\text{Prop } \alpha P'$

proof –

from *Trans have distinct(bn alpha)*

by(*auto dest: boundOutputDistinct*)

note $\text{facts} = \langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{subject } \alpha \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# P' \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

have $\text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P')$ **by** *simp*

note *Trans*

moreover have $\text{length } [] = \text{inputLength}((\nu x)P)$ **and** *distinct []*

by(*auto simp add: inputLength-inputLength'-inputLength''.simps*)

moreover have $\text{length } [] = \text{inputLength}((\nu x)P)$ **and** *distinct []*

by(*auto simp add: inputLength-inputLength'-inputLength''.simps*)

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

moreover note $\langle \text{length}(\text{bn } \alpha) = \text{residualLength}(\alpha \prec P') \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$

ultimately show *?thesis using facts*

proof(*cases rule: semanticsCases[of - - - - - C x x x x]*)

case (*cOpen P M xvec y yvec N P'*)

moreover then have $y \in \text{supp } ((x, y) \cdot N)$ **using** *facts*

apply(*clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqts*)

apply(*drule pt-set-bij2* [**where** $\text{pi} = [(x, y)]$, **where** $x = x$, *OF pt-name-inst*, *OF*

```

at-name-inst])
  by(auto simp add: calc-atm eqvts fresh-def)
  ultimately show ?thesis using facts
  apply(clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts)
  by(rule rOpen) (auto simp add: residualInject boundOutputApp)
next
case (cBrOpen P M xvec y yvec N P')
moreover then have  $y \in \text{supp } ([ (x, y) ] \cdot N)$  using facts
  apply(clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts)
  apply(drule pt-set-bij2[where pi=[(x, y)], where x=x, OF pt-name-inst, OF at-name-inst])
  by(auto simp add: calc-atm eqvts fresh-def)
  ultimately show ?thesis using facts
  apply(clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts)
  by(rule rBrOpen) (auto simp add: residualInject boundOutputApp)
next
qed (auto simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts
  intro: rScope rBrClose)
qed

```

lemma *resCases'*[consumes 7, case-names cOpen cBrOpen cRes cBrClose]:

```

fixes  $\Psi$     :: 'b
and x      :: name
and P     :: ('a, 'b, 'c) psi
and  $\alpha$   :: 'a action
and P'    :: ('a, 'b, 'c) psi
and C     :: 'f::fs-name

assumes Trans:  $\Psi \triangleright (\nu x)P \mapsto \alpha \prec P'$ 
and x #  $\Psi$ 
and x #  $\alpha$ 
and x # P'
and bn  $\alpha$  #*  $\Psi$ 
and bn  $\alpha$  #* P
and bn  $\alpha$  #* subject  $\alpha$ 
and rOpen:  $\bigwedge M \text{ xvec yvec } y N P'. [\Psi \triangleright ([ (x, y) ] \cdot P) \mapsto M(\nu*(\text{xvec}@yvec))\langle N \rangle \prec P'; y \in \text{supp } N;$ 

$$x \# N; x \# P'; x \neq y; y \# \text{xvec}; y \# \text{yvec}; y \# M;$$

distinct xvec; distinct yvec;

$$\text{xvec} \#* \Psi; y \# \Psi; \text{yvec} \#* \Psi; \text{xvec} \#* P; y \# P;$$

yvec #* P; xvec #* M; y # M;

$$\text{yvec} \#* M; \text{xvec} \#* \text{yvec}] \implies$$


$$\text{Prop } (M(\nu*(\text{xvec}@y\#\text{yvec}))\langle N \rangle) P'$$

and rBrOpen:  $\bigwedge M \text{ xvec yvec } y N P'. [\Psi \triangleright ([ (x, y) ] \cdot P) \mapsto {}_i M(\nu*(\text{xvec}@yvec))\langle N \rangle \prec P'; y \in \text{supp } N;$ 

```

$x \# N; x \# P'; x \neq y; y \# xvec; y \# yvec; y \# M;$
distinct xvec; distinct yvec;
 $xvec \#* \Psi; y \# \Psi; yvec \#* \Psi; xvec \#* P; y \# P;$
 $yvec \#* P; xvec \#* M; y \# M;$
 $yvec \#* M; xvec \#* yvec \implies$
 $Prop (iM(\nu*(xvec@y\#yvec))\langle N \rangle) P'$
and *rScope*: $\bigwedge P'. [\Psi \triangleright P \mapsto \alpha \prec P'] \implies Prop \alpha (\nu x) P'$
and *rBrClose*: $\bigwedge M \ xvec \ N \ P'.$
 $[\Psi \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P';$
 $x \in supp \ M;$
 $distinct \ xvec; xvec \#* \Psi; xvec \#* P;$
 $xvec \#* M;$
 $x \# \Psi; x \# xvec;$
 $xvec \#* C] \implies Prop (\tau) ((\nu x)((\nu*xvec)P')$

shows $Prop \alpha P'$

proof –

from *Trans* **have** $distinct(bn \ \alpha)$
by (*auto dest: boundOutputDistinct*)
note *facts* = $\langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* P \rangle \langle bn \ \alpha \#* subject \ \alpha \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \#$
 $P' \rangle \langle distinct(bn \ \alpha) \rangle$
have $length(bn \ \alpha) = residualLength(\alpha \prec P')$
by *simp*
note *Trans*
moreover **have** $length \ [] = inputLength((\nu x)P)$ **and** $distinct \ []$
by (*auto simp add: inputLength-inputLength'-inputLength''.simps*)
moreover **have** $length \ [] = inputLength((\nu x)P)$ **and** $distinct \ []$
by (*auto simp add: inputLength-inputLength'-inputLength''.simps*)
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
moreover **note** $\langle length(bn \ \alpha) = residualLength(\alpha \prec P') \rangle \langle distinct(bn \ \alpha) \rangle$
ultimately **show** *?thesis using facts*
proof (*cases rule: semanticsCases[of - - - - - C x x x x]*)
case (*cOpen P M xvec y yvec N P'*)
moreover **then** **have** $y \in supp \ ((x, y) \cdot N)$ **using** *facts*
apply (*clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply (*drule pt-set-bij2[where pi=[(x, y)], OF pt-name-inst, OF at-name-inst]*)
by (*auto simp add: calc-atm eqvts fresh-def*)
ultimately **show** *?thesis using facts*
apply (*clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts*)
apply (*rule rOpen*) — 20 subgoals
apply (*drule semantics.eqvt[where pi=[(x, y)]]*)
apply (*auto simp add: eqvts residualInject boundOutputApp*)

```

done
next
case (cBrOpen P M xvec y yvec N P')
moreover then have  $y \in \text{supp } ([x, y] \cdot N)$  using facts
  apply (clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts)
  apply (drule pt-set-bij2[where pi=[(x, y)], OF pt-name-inst, OF at-name-inst])
  by (auto simp add: calc-atm eqvts fresh-def)
ultimately show ?thesis using facts
  apply (clarsimp simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts)
  apply (rule rBrOpen) — 20 subgoals
    apply (drule semantics.eqvt[where pi=[(x, y)])
    apply (auto simp add: eqvts residualInject boundOutputApp)
done
qed (auto simp add: psi.inject alpha abs-fresh residualInject boundOutputApp boundOutput.inject eqvts
  intro: rScope rBrClose)
qed

```

lemma *resInputCases'*[consumes 4, case-names cRes]:

```

fixes  $\Psi$    :: 'b
and x     :: name
and M     :: 'a
and N     :: 'a
and P     :: ('a, 'b, 'c) psi
and R     :: ('a, 'b, 'c) psi
and C     :: 'f::fs-name

```

assumes *Trans*: $\Psi \triangleright (\nu x)P \mapsto M(N) \prec R$

```

and 2:  $x \# \Psi$ 
and  $x \# (M(N))$ 
and 4:  $x \# R$ 
and rScope:  $\bigwedge P'. [\Psi \triangleright P \mapsto M(N) \prec P] \implies \text{Prop } ((\nu x)P')$ 

```

shows *Prop R*

proof —

from *Trans* **obtain** α **where** 1: $\Psi \triangleright (\nu x)P \mapsto \alpha \prec R$ **and** 5: $bn \alpha \# \Psi$ **and** 6: $bn \alpha \# P$ **and** 7: $bn \alpha \# \text{subject } \alpha$ **and** $\alpha = M(N)$ **by** *auto*

from $\langle x \# (M(N)) \rangle \langle \alpha = (M(N)) \rangle$ **have** 3: $x \# \alpha$ **by** *simp*

show ?thesis **using** 1 2 3 4 5 6 7 $\langle \alpha = M(N) \rangle$ *rScope*

by (*induct rule: resCases'*) (*auto simp add: residualInject*)

qed

lemma *resBrInputCases'*[consumes 4, case-names cRes]:

```

fixes  $\Psi$    :: 'b
and x     :: name
and M     :: 'a
and N     :: 'a

```

and P :: ('a, 'b, 'c) psi
and R :: ('a, 'b, 'c) psi
and C :: 'f::fs-name

assumes $Trans$: $\Psi \triangleright (\nu x)P \mapsto \iota M(N) \prec R$
and 2: $x \# \Psi$
and $x \# (\iota M(N))$
and 4: $x \# R$
and $rScope$: $\bigwedge P'. [\Psi \triangleright P \mapsto \iota M(N) \prec P'] \implies Prop ((\nu x)P')$

shows $Prop R$

proof –

from $Trans$ **obtain** α **where** 1: $\Psi \triangleright (\nu x)P \mapsto \alpha \prec R$ **and** 5: $bn \alpha \#* \Psi$ **and**
 6: $bn \alpha \#* P$ **and** 7: $bn \alpha \#*$ subject α **and** $\alpha = \iota M(N)$ **by** *auto*

from $\langle x \# (\iota M(N)) \rangle \langle \alpha = (\iota M(N)) \rangle$ **have** 3: $x \# \alpha$ **by** *simp*

show ?thesis **using** 1 2 3 4 5 6 7 $\langle \alpha = \iota M(N) \rangle$ $rScope$

by(*induct rule: resCases*) (*auto simp add: residualInject*)

qed

lemma $resOutputCases''$ [*consumes 7, case-names cOpen cRes*]:

fixes Ψ :: 'b
and x :: name
and M :: 'a
and N :: 'a
and P :: ('a, 'b, 'c) psi
and R :: ('a, 'b, 'c) psi
and C :: 'f::fs-name

assumes $Trans$: $\Psi \triangleright (\nu x)P \mapsto M(\nu*(zvec1 @ zvec2))\langle N \rangle \prec R$

and 1: $x \# \Psi$

and $x \# (M(\nu*(zvec1 @ zvec2))\langle N \rangle)$

and 3: $x \# R$

and $(zvec1 @ zvec2) \#* \Psi$

and $(zvec1 @ zvec2) \#* P$

and $(zvec1 @ zvec2) \#* M$

and $rOpen$: $\bigwedge M \ xvec \ yvec \ y \ N \ P'. [\Psi \triangleright ((x, y) \cdot P) \mapsto M(\nu*(xvec@yvec))\langle N \rangle \prec P'; y \in supp \ N;$

$x \# N; x \# P'; x \neq y; y \# xvec; y \# yvec; y \# M;$

$distinct \ xvec; distinct \ yvec;$

$xvec \#* \Psi; y \# \Psi; yvec \#* \Psi; xvec \#* P; y \# P;$

$yvec \#* P; xvec \#* M; y \# M;$

$yvec \#* M; xvec \#* yvec] \implies$

$Prop \ P'$

and $rScope$: $\bigwedge P'. [\Psi \triangleright P \mapsto M(\nu*(zvec1 @ zvec2))\langle N \rangle \prec P'] \implies Prop ((\nu x)P')$

shows $Prop R$

proof –

from $Trans$ **have** $distinct \ (zvec1 @ zvec2)$ **by**(*auto dest: boundOutputDistinct*)

obtain α **where** $\alpha = M(\nu^*(zvec1 @ zvec2)) \langle N \rangle$ **by** *simp*
with *Trans* $\langle (zvec1 @ zvec2) \#^* \Psi \rangle \langle (zvec1 @ zvec2) \#^* P \rangle \langle (zvec1 @ zvec2) \#^* M \rangle$
have α *Trans*: $\Psi \triangleright (\nu x)P \mapsto \alpha \prec R$ **and** 4: $bn \ \alpha \ \#^* \ \Psi$ **and** 5: $bn \ \alpha \ \#^* \ P$ **and**
6: $bn \ \alpha \ \#^*$ *subject* α
by *simp+*
from $\langle x \# (M(\nu^*(zvec1 @ zvec2)) \langle N \rangle) \rangle \langle \alpha = M(\nu^*(zvec1 @ zvec2)) \langle N \rangle \rangle$ **have**
2: $x \# \alpha$ **by** *simp*
show *?thesis* **using** α *Trans* 1 2 3 4 5 6 $\langle \alpha = M(\nu^*(zvec1 @ zvec2)) \langle N \rangle \rangle$ *rOpen*
rScope
proof(*induct rule: resCases'*[**where** $C = (zvec1, zvec2, C)$])
case *cBrOpen*
then show *?case*
by(*auto simp add: residualInject boundOutputApp*)
next
case *cRes*
then show *?case*
by(*auto simp add: residualInject boundOutputApp*)
next
case *cBrClose*
then show *?case*
by(*auto simp add: residualInject boundOutputApp*)
next
case(*cOpen M' xvec yvec y N' P'*)
then show *?case*
by(*auto simp add: residualInject boundOutputApp*)
qed
qed

lemma *resOutputCases''*[*consumes* 7, *case-names* *cOpen cRes*]:

fixes Ψ **::** 'b
and x **::** *name*
and z **::** *name*
and M **::** 'a
and N **::** 'a
and P **::** ('a, 'b, 'c) *psi*
and R **::** ('a, 'b, 'c) *psi*
and C **::** 'f::*fs-name*

assumes *Trans*: $\Psi \triangleright (\nu x)P \mapsto M(\nu^*(zvec1 @ y \# zvec2)) \langle N \rangle \prec R$

and 1: $x \# \Psi$
and $x \# (M(\nu^*(zvec1 @ y \# zvec2)) \langle N \rangle)$
and 3: $x \# R$
and $(zvec1 @ y \# zvec2) \#^* \Psi$
and $(zvec1 @ y \# zvec2) \#^* P$
and $(zvec1 @ y \# zvec2) \#^* M$
and *rOpen*: $\bigwedge M \ xvec \ yvec \ y \ N \ P'. \llbracket \Psi \triangleright ((x, y) \cdot P) \mapsto M(\nu^*(xvec @ yvec)) \langle N \rangle \prec P'; y \in \text{supp } N; \right.$
 $\left. x \# N; x \# P'; x \neq y; y \# xvec; y \# yvec; y \# M; \right.$

distinct xvec; distinct yvec;
xvec # Ψ; y # Ψ; yvec #* Ψ; xvec #* P; y # P;*
yvec # P; xvec #* M; y # M;*
yvec # M; xvec #* yvec]] ==>*
Prop P'
and *rScope: ⋀P'. [[Ψ ▷ P ⟶ M(ν*(zvec1@y#zvec2))⟨N⟩ < P]] ==> Prop*
((νx)P')

shows *Prop R*

proof –

from *Trans have distinct (zvec1@y#zvec2) by(auto dest: boundOutputDistinct)*
obtain *α where α=M(ν*(zvec1@y#zvec2))⟨N⟩ by simp*
with *Trans ⟨(zvec1@y#zvec2) #* Ψ⟩ ⟨(zvec1@y#zvec2) #* P⟩ ⟨(zvec1@y#zvec2)*
M⟩*

have *αTrans: Ψ ▷ (νx)P ⟶ α < R and 4: bn α #* Ψ and 5: bn α #* P and*
6: bn α # subject α*

by *simp+*

from *⟨x # (M(ν*(zvec1@y#zvec2))⟨N⟩)⟩ ⟨α=M(ν*(zvec1@y#zvec2))⟨N⟩⟩ have*
2: x # α by simp

show *?thesis using αTrans 1 2 3 4 5 6 ⟨α=M(ν*(zvec1@y#zvec2))⟨N⟩⟩ rOpen*
rScope

proof(*induct rule: resCases'[where C=(zvec1, zvec2, z, C)]*)

case *cBrOpen*

then show *?case*

by(*auto simp add: residualInject boundOutputApp*)

next

case *cRes*

then show *?case*

by(*auto simp add: residualInject boundOutputApp*)

next

case *cBrClose*

then show *?case*

by(*auto simp add: residualInject boundOutputApp*)

next

case(*cOpen M' xvec yvec y N' P'*)

then show *?case*

by(*auto simp add: residualInject boundOutputApp*)

qed

qed

abbreviation

statImpJudge (- ↔ - [80, 80] 80)

where *Ψ ↔ Ψ' ≡ Assertion.StatImp Ψ Ψ'*

lemma *statEqTransition:*

fixes *Ψ :: 'b*

and *P :: ('a, 'b, 'c) psi*

and *Rs :: ('a, 'b, 'c) residual*

and *Ψ' :: 'b*

```

assumes  $\Psi \triangleright P \mapsto Rs$ 
and  $\Psi \simeq \Psi'$ 

shows  $\Psi' \triangleright P \mapsto Rs$ 
using assms
proof(nominal-induct avoiding:  $\Psi'$  rule: semantics.strong-induct)
  case(cInput  $\Psi M K xvec N Tvec P \Psi'$ )
    from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi' \vdash M \leftrightarrow K$ 
      by(simp add: AssertionStatImp-def AssertionStatEq-def)
    then show ?case using  $\langle distinct\ xvec \rangle \langle set\ xvec \subseteq\ supp\ N \rangle \langle length\ xvec = length\ Tvec \rangle$ 
      by(rule Input)
  next
    case(cBrInput  $\Psi K M xvec N Tvec P \Psi'$ )
      from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi \vdash K \succeq M \rangle$  have  $\Psi' \vdash K \succeq M$ 
        by(simp add: AssertionStatImp-def AssertionStatEq-def)
      then show ?case using  $\langle distinct\ xvec \rangle \langle set\ xvec \subseteq\ supp\ N \rangle \langle length\ xvec = length\ Tvec \rangle$ 
        by(rule BrInput)
  next
    case(Output  $\Psi M K N P \Psi'$ )
      from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi' \vdash M \leftrightarrow K$ 
        by(simp add: AssertionStatImp-def AssertionStatEq-def)
      then show ?case by (rule semantics.Output)
  next
    case(BrOutput  $\Psi M K N P \Psi'$ )
      from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi \vdash M \preceq K \rangle$  have  $\Psi' \vdash M \preceq K$ 
        by(simp add: AssertionStatImp-def AssertionStatEq-def)
      then show ?case by (rule semantics.BrOutput)
  next
    case(Case  $\Psi P Rs \varphi Cs \Psi'$ )
      then have  $\Psi' \triangleright P \mapsto Rs$  by(intro Case)
      moreover note  $\langle \varphi, P \rangle \in set\ Cs$ 
      moreover from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi \vdash \varphi \rangle$  have  $\Psi' \vdash \varphi$ 
        by(simp add: AssertionStatImp-def AssertionStatEq-def)
      ultimately show ?case using  $\langle guarded\ P \rangle$  by (rule semantics.Case)
  next
    case(cPar1  $\Psi \Psi Q P \alpha P' xvec Q \Psi'$ )
      then show ?case
      by(intro Par1 (auto intro: Composition))
  next
    case(cPar2  $\Psi \Psi P Q \alpha Q' xvec P \Psi'$ )
      then show ?case
      by(intro Par2 (auto intro: Composition))
  next
    case(cComm1  $\Psi \Psi Q P M N P' xvec \Psi P Q K yvec Q' yvec \Psi'$ )
      then show ?case
      by(clarsimp, intro Comm1 (blast intro: Composition statEqEnt))

```



```

next
  case(cComm2  $\Psi \Psi_Q P M zvec N P' xvec \Psi P Q K Q' yvec \Psi'$ )
  then show ?case
    by(clarsimp, intro Comm2) (blast intro: Composition statEqEnt)+
next
  case(cBrMerge  $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q \Psi'$ )
  then show ?case
    by(clarsimp, intro BrMerge) (blast intro: Composition)+
next
  case(cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q \Psi'$ )
  then show ?case
    by(clarsimp, intro BrComm1) (blast intro: Composition)+
next
  case(cBrComm2  $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q Q' A_Q \Psi'$ )
  then show ?case
    by(clarsimp, intro BrComm2) (blast intro: Composition)+
next
  case(cBrClose  $\Psi P M xvec N P' x \Psi'$ )
  then show ?case by(force intro: BrClose)
next
  case(cOpen  $\Psi P M xvec N P' x yvec \Psi'$ )
  then show ?case by(force intro: Open)
next
  case(cBrOpen  $\Psi P M xvec N P' x yvec \Psi'$ )
  then show ?case by(force intro: BrOpen)
next
  case(cScope  $\Psi P \alpha P' x \Psi'$ )
  then show ?case by(force intro: Scope)
next
  case(Bang  $\Psi P Rs \Psi'$ )
  then show ?case by(force intro: semantics.Bang)
qed

```

lemma *brInputTermSupp*:

```

fixes  $\Psi :: 'b::fs-name$ 
  and  $P :: ('a, 'b, ('c::fs-name)) psi$ 
  and  $P' :: ('a, 'b, 'c) psi$ 
  and  $N :: 'a::fs-name$ 
  and  $K :: 'a::fs-name$ 

```

assumes $\Psi \triangleright P \mapsto \downarrow K(N) \prec P'$

```

shows  $(supp K) \subseteq ((supp P)::name set)$ 
using assms
proof(nominal-induct rule: brInputInduct)
  case(cBrInput  $\Psi K M xvec N Tvec P$ )
  from  $\langle \Psi \vdash K \succeq M \rangle$  have  $(supp K) \subseteq ((supp M)::name set)$ 
  by(simp add: chanInConSupp)
  then show ?case

```

```

    by (metis Un-commute Un-upper2 psi.supp(3) subset-trans)
next
case(cCase  $\Psi P M N P' \varphi Cs$ )
then have  $\text{supp } M \subseteq ((\text{supp } P)::\text{name set})$ 
  by simp
from  $\langle (\varphi, P) \in \text{set } Cs \rangle$ 
have  $\{(\varphi, P)\} \subseteq \text{set } Cs$ 
  by simp
moreover have  $\text{finite } \{(\varphi, P)\}$  by simp
moreover have  $\text{finite } (\text{set } Cs)$  by simp
ultimately have  $((\text{supp } \{(\varphi, P)\})::\text{name set}) \subseteq ((\text{supp } (\text{set } Cs))::\text{name set})$ 
  by(simp add: supp-subset)

moreover have  $\text{supp } \{(\varphi, P)\} = ((\text{supp } (\varphi, P))::\text{name set})$ 
  by (meson supp-singleton)
moreover have  $\text{supp } P \subseteq \text{supp } (\varphi, P)$ 
  by (metis Un-upper2 supp-prod)
ultimately have  $((\text{supp } P)::\text{name set}) \subseteq ((\text{supp } Cs)::\text{name set})$ 
  by (auto simp add: supp-list-set)

moreover have  $((\text{supp } Cs)::\text{name set}) = ((\text{supp } (\text{Cases } Cs))::\text{name set})$ 
  unfolding psi.supp
  apply(induct rule: psiCases.induct)
  apply(metis psiCase.supp(1) psiCases.simps(1) set-empty2 supp-list-nil)
  by simp (metis Un-assoc psiCase.supp(2) supp-list-cons supp-prod)

ultimately have  $((\text{supp } P)::\text{name set}) \subseteq ((\text{supp } (\text{Cases } Cs))::\text{name set})$ 
  by simp

with  $\langle \text{supp } M \subseteq \text{supp } P \rangle$ 
show ?case by simp
next
case(cPar1  $\Psi \Psi_Q P M N P' A_Q Q$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } P)::\text{name set})$ 
  by auto
then show ?case
  by(auto simp add: psi.supp)
next
case(cPar2  $\Psi \Psi_P Q M N Q' A_P P$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } Q)::\text{name set})$ 
  by auto
then show ?case
  by(auto simp add: psi.supp)
next
case(cBrMerge  $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q$ )
then show ?case
  by(auto simp add: psi.supp)
next
case(cScope  $\Psi P M N P' x$ )

```

```

then have ((supp M)::name set)  $\subseteq$  ((supp P)::name set)
  by simp
then show ?case
  by(simp add: psi.supp) (metis abs-supp cScope.hyps fresh-def insert-Diff-single
subset-insert-iff)
next
  case(cBang  $\Psi$  P M N P')
  then show ?case
  by(simp add: psi.supp)
qed

```

lemma *brOutputTermSupp*:

```

fixes  $\Psi$  :: 'b::fs-name
  and P :: ('a, 'b, ('c::fs-name)) psi
  and P' :: ('a, 'b, 'c) psi
  and N :: 'a::fs-name
  and K :: 'a::fs-name
  and xvec :: name list

```

assumes $\Psi \triangleright P \mapsto RBrOut\ K\ ((\nu*xvec)N \prec' P')$

```

shows (supp K)  $\subseteq$  ((supp P)::name set)
  using assms
proof(nominal-induct rule: brOutputInduct)
  case(cBrOutput  $\Psi$  M K N P)
  from  $\langle \Psi \vdash M \preceq K \rangle$  have (supp K)  $\subseteq$  ((supp M)::name set)
    by(simp add: chanOutConSupp)
  then show ?case
    by (metis Un-commute Un-upper2 psi.supp(2) subset-iff-psubset-eq subset-trans)
next
  case(cCase  $\Psi$  P M B  $\varphi$  Cs)
  then have supp M  $\subseteq$  ((supp P)::name set)
    by simp
  from  $\langle (\varphi, P) \in set\ Cs \rangle$ 
  have  $\{(\varphi, P)\} \subseteq set\ Cs$ 
    by simp
  moreover have finite  $\{(\varphi, P)\}$  by simp
  moreover have finite (set Cs) by simp
  ultimately have ((supp  $\{(\varphi, P)\}$ )::name set)  $\subseteq$  ((supp (set Cs))::name set)
    by(simp add: supp-subset)

  moreover have supp  $\{(\varphi, P)\} = ((supp\ (\varphi, P))::name\ set)$ 
    by(meson supp-singleton)
  moreover have supp P  $\subseteq$  supp  $(\varphi, P)$ 
    by (metis Un-upper2 supp-prod)
  ultimately have ((supp P)::name set)  $\subseteq$  ((supp Cs)::name set)
    by (auto simp add: supp-list-set)

```

moreover have ((*supp* *Cs*)::*name set*) = ((*supp* (*Cases* *Cs*))::*name set*)

```

unfolding psi.supp
apply(induct rule: psiCases.induct)
apply(metis psiCase.supp(1) psiCases.simps(1) set-empty2 supp-list-nil)
by simp (metis Un-assoc psiCase.supp(2) supp-list-cons supp-prod)

ultimately have  $((\text{supp } P)::\text{name set}) \subseteq ((\text{supp } (\text{Cases } Cs))::\text{name set})$ 
by simp

with  $\langle \text{supp } M \subseteq \text{supp } P \rangle$ 
show ?case by simp
next
case(cPar1  $\Psi \Psi_Q P M \text{xvec } N P' A_Q Q$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } P)::\text{name set})$ 
by auto
then show ?case
by(auto simp add: psi.supp)
next
case(cPar2  $\Psi \Psi_P Q M \text{xvec } N Q' A_P P$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } Q)::\text{name set})$ 
by auto
then show ?case
by(auto simp add: psi.supp)
next
case(cBrComm1  $\Psi \Psi_Q P M N P' A_P \Psi_P Q \text{xvec } Q' A_Q$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } Q)::\text{name set})$ 
by auto
then show ?case
by(auto simp add: psi.supp)
next
case(cBrComm2  $\Psi \Psi_Q P M \text{xvec } N P' A_P \Psi_P Q Q' A_Q$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } P)::\text{name set})$ 
by auto
then show ?case
by(auto simp add: psi.supp)
next
case(cBrOpen  $\Psi P M \text{xvec } yvec N P' x$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } P)::\text{name set})$ 
by simp
with  $\langle x \# M \rangle$ 
show ?case
by(simp add: psi.supp) (metis abs-supp cBrOpen.hyps fresh-def insert-Diff-single subset-insert-iff)
next
case(cScope  $\Psi P M \text{xvec } N P' x$ )
then have  $((\text{supp } M)::\text{name set}) \subseteq ((\text{supp } P)::\text{name set})$ 
by simp
then show ?case
by(simp add: psi.supp) (metis abs-supp cScope.hyps fresh-def insert-Diff-single subset-insert-iff)

```

```

next
  case cBang
  then show ?case
    by(simp add: psi.supp)
qed

```

```

lemma actionPar1Dest':
  fixes  $\alpha :: ('a::fs-name) action$ 
  and  $P :: ('a, 'b::fs-name, 'c::fs-name) psi$ 
  and  $\beta :: ('a::fs-name) action$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $R :: ('a, 'b, 'c) psi$ 

```

```

assumes  $\alpha \prec P = \beta \prec (Q \parallel R)$ 
  and  $bn \alpha \#* R$ 
  and  $bn \beta \#* R$ 

```

```

obtains  $T$  where  $P = T \parallel R$  and  $\alpha \prec T = \beta \prec Q$ 
  using assms
  apply(cases rule: actionCases[where  $\alpha=\alpha$ ])
    apply (metis residualInject'(1))
    apply (metis residualInject'(7))
    apply (smt (z3) bn.simps(3) boundOutputPar1Dest create-residual.simps(3))
residualInject'(8))
    apply (smt (z3) bn.simps(4) boundOutputPar1Dest create-residual.simps(4))
residualInject'(9))
  by (metis residualInject'(10))

```

```

lemma actionPar2Dest':
  fixes  $\alpha :: ('a::fs-name) action$ 
  and  $P :: ('a, 'b::fs-name, 'c::fs-name) psi$ 
  and  $\beta :: ('a::fs-name) action$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $R :: ('a, 'b, 'c) psi$ 

```

```

assumes  $\alpha \prec P = \beta \prec (Q \parallel R)$ 
  and  $bn \alpha \#* Q$ 
  and  $bn \beta \#* Q$ 

```

```

obtains  $T$  where  $P = Q \parallel T$  and  $\alpha \prec T = \beta \prec R$ 
  using assms
  apply(cases rule: actionCases[where  $\alpha=\alpha$ ])
    apply (metis residualInject'(1))
    apply (metis residualInject'(7))
    apply (smt (z3) bn.simps(3) boundOutputPar2Dest create-residual.simps(3))
residualInject'(8))
    apply (smt (z3) bn.simps(4) boundOutputPar2Dest create-residual.simps(4))
residualInject'(9))
  by (metis residualInject'(10))

```

lemma *expandNonTauFrame*:

fixes Ψ $:: 'b$
and P $:: ('a, 'b, 'c)$ *psi*
and α $:: 'a$ *action*
and P' $:: ('a, 'b, 'c)$ *psi*
and A_P $::$ *name list*
and Ψ_P $:: 'b$
and C $:: 'f::fs-name$
and C' $:: 'g::fs-name$

assumes *Trans*: $\Psi \triangleright P \mapsto \alpha \prec P'$

and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $bn \alpha \#* subject \alpha$
and $A_P \#* P$
and $A_P \#* \alpha$
and $A_P \#* C$
and $A_P \#* C'$
and $bn \alpha \#* P$
and $bn \alpha \#* C'$
and $\alpha \neq \tau$

obtains $p \Psi' A_P' \Psi_P'$ **where** $set p \subseteq set(bn \alpha) \times set(bn(p \cdot \alpha))$ **and** $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'$ **and** *distinctPerm* p **and**

extractFrame $P' = \langle A_P', \Psi_P' \rangle$ **and** $A_P' \#* P'$ **and** $A_P' \#* \alpha$ **and** $A_P' \#* (p \cdot \alpha)$ **and**

$A_P' \#* C$ **and** $(bn(p \cdot \alpha)) \#* C'$ **and** $(bn(p \cdot \alpha)) \#* \alpha$ **and** $(bn(p \cdot \alpha)) \#* P'$ **and** *distinct* A_P'

proof –

assume $A: \bigwedge p \Psi' \Psi_P' A_P'$.

$\llbracket set p \subseteq set(bn \alpha) \times set(bn(p \cdot \alpha)); (p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'; distinctPerm p;$
extractFrame $P' = \langle A_P', \Psi_P' \rangle; A_P' \#* P'; A_P' \#* \alpha; A_P'$

$\#* (p \cdot \alpha);$

$A_P' \#* C; (bn(p \cdot \alpha)) \#* C'; (bn(p \cdot \alpha)) \#* \alpha; (bn(p \cdot \alpha))$

$\#* P'; distinct A_P' \rrbracket$

$\implies thesis$

from *Trans* **have** *distinct*($bn \alpha$) **by**(*auto dest: boundOutputDistinct*)

with *Trans* $\langle bn \alpha \#* subject \alpha \rangle \langle A_P \#* P \rangle \langle A_P \#* \alpha \rangle$ **have** $A_P \#* P'$

by(*drule-tac freeFreshChainDerivative*) *auto*

{
fix V $:: 'a$ *list*
and W $:: ('a$ *action)* *list*
and X $::$ *name list*
and Y $:: 'b$ *list*
and Z $:: ('a, 'b, 'c)$ *psi list*

assume $bn\ \alpha \#* V$ **and** $bn\ \alpha \#* W$ **and** $bn\ \alpha \#* X$ **and** $bn\ \alpha \#* Y$ **and** $bn\ \alpha \#* Z$ **and** $A_P \#* V$ **and** $A_P \#* W$ **and** $A_P \#* X$ **and** $A_P \#* Y$ **and** $A_P \#* Z$

with *assms* **obtain** $p\ \Psi' A_{P'} \Psi_{P'}$ **where** $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **and** $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$ **and** *distinctPerm* p **and** *extractFrame* $P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** $A_{P'} \#* P'$ **and** $A_{P'} \#* \alpha$ **and** $A_{P'} \#* (p \cdot \alpha)$ **and** $A_{P'} \#* C$ **and** $(bn(p \cdot \alpha)) \#* C'$ **and** $(bn(p \cdot \alpha)) \#* \alpha$ **and** $(bn(p \cdot \alpha)) \#* P'$ **and** $A_{P'} \#* V$ **and** $A_{P'} \#* W$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* Z$ **and** *distinct* $A_{P'}$ **and** $(bn(p \cdot \alpha)) \#* V$ **and** $(bn(p \cdot \alpha)) \#* W$ **and** $(bn(p \cdot \alpha)) \#* X$ **and** $(bn(p \cdot \alpha)) \#* Y$ **and** $(bn(p \cdot \alpha)) \#* Z$ **using** $\langle A_{P'} \#* P' \rangle$ $\langle distinct(bn\ \alpha) \rangle$

proof(*nominal-induct* $\Psi\ P\ Rs==\alpha \prec P' A_P \Psi_P$ *avoiding: C C' \alpha P' V W X Y Z* *arbitrary: thesis rule: semanticsFrameInduct*)

case(*cAlpha* $\Psi\ P\ A_P\ \Psi_P\ p\ C\ C'\ \alpha\ P'\ V\ W\ X\ Y\ Z$)

then **obtain** $q\ \Psi' A_{P'} \Psi_{P'}$ **where** $Sq: set\ q \subseteq set(bn\ \alpha) \times set(bn(q \cdot \alpha))$ **and** $PeqP': (q \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$ **and** *distinctPerm* q **and** *FrP'*: *extractFrame* $P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** $A_{P'} \#* P'$ **and** $A_{P'} \#* \alpha$ **and** $A_{P'} \#* (q \cdot \alpha)$ **and** $A_{P'} \#* C$ **and** $(bn(q \cdot \alpha)) \#* C'$ **and** $(bn(q \cdot \alpha)) \#* \alpha$ **and** $(bn(q \cdot \alpha)) \#* P'$ **and** $A_{P'} \#* V$ **and** $A_{P'} \#* W$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* Z$ **and** *distinct* $A_{P'}$ **and** $(bn(q \cdot \alpha)) \#* V$ **and** $(bn(q \cdot \alpha)) \#* W$ **and** $(bn(q \cdot \alpha)) \#* X$ **and** $(bn(q \cdot \alpha)) \#* Y$ **and** $(bn(q \cdot \alpha)) \#* Z$ **by** *metis*

have $Sp: set\ p \subseteq set\ A_P \times set\ (p \cdot A_P)$ **by** *fact*

from Sq **have** $(p \cdot set\ q) \subseteq p \cdot (set(bn\ \alpha) \times set(bn(q \cdot \alpha)))$ **by**(*simp add: subsetClosed*)

then **have** $set(p \cdot q) \subseteq set(bn(p \cdot \alpha)) \times set(p \cdot bn(q \cdot \alpha))$ **by**(*simp add: eqvts*)

with $\langle A_P \#* \alpha \rangle$ $\langle (p \cdot A_P) \#* \alpha \rangle$ Sp **have** $set(p \cdot q) \subseteq set(bn\ \alpha) \times set(bn((p \cdot q) \cdot \alpha))$ **by**(*simp add: perm-compose bnEqvt[symmetric]*)

moreover **from** $PeqP'$ **have** $(p \cdot (q \cdot \Psi_P) \otimes \Psi') \simeq (p \cdot \Psi_{P'})$ **by**(*simp add: AssertionStatEqClosed*)

then **have** $((p \cdot q) \cdot p \cdot \Psi_P) \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_{P'})$ **apply**(*subst perm-compose[symmetric]*) **by**(*simp add: eqvts*)

moreover **from** $\langle distinctPerm\ q \rangle$ **have** *distinctPerm* $(p \cdot q)$ **by** *simp*

moreover **from** $\langle (bn(q \cdot \alpha)) \#* C' \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot C')$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle A_P \#* \alpha \rangle$ $\langle (p \cdot A_P) \#* \alpha \rangle$ $\langle A_P \#* C' \rangle$ $\langle (p \cdot A_P) \#* C' \rangle$ Sp **have** $bn((p$

$\cdot q) \cdot \alpha) \#* C'$
by(*simp add: perm-compose bnEqvt[symmetric]*)
moreover from $\text{Fr}P'$ **have** $(p \cdot \text{extractFrame } P') = p \cdot \langle A_{P'}, \Psi_{P'} \rangle$ **by** *simp*
with $\langle A_P \#* P' \rangle \langle (p \cdot A_P) \#* P' \rangle$ *Sp* **have** $\text{extractFrame } P' = \langle p \cdot A_{P'}, p \cdot \Psi_{P'} \rangle$
by(*simp add: eqvts*)
moreover from $\langle A_{P'} \#* P' \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot P')$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* P' \rangle \langle (p \cdot A_P) \#* P' \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* P'$ **by** *simp*
moreover from $\langle A_{P'} \#* \alpha \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot \alpha)$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* \alpha$ **by** *simp*
moreover from $\langle A_{P'} \#* C \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot C)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* C \rangle \langle (p \cdot A_P) \#* C \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* C$ **by** *simp*
moreover from $\langle (bn(q \cdot \alpha)) \#* \alpha \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot \alpha)$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle$ *Sp* **have** $bn((p \cdot q) \cdot \alpha) \#* \alpha$
by(*simp add: perm-compose eqvts*)
moreover from $\langle (bn(q \cdot \alpha)) \#* P' \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot P')$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* P' \rangle \langle (p \cdot A_P) \#* P' \rangle$ *Sp* **have** $bn((p \cdot q) \cdot \alpha) \#* P'$
by(*simp add: perm-compose eqvts*)
moreover from $\langle A_{P'} \#* (q \cdot \alpha) \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot q \cdot \alpha)$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with *Sp* $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle$ **have** $(p \cdot A_{P'}) \#* ((p \cdot q) \cdot \alpha)$
by(*simp add: perm-compose*)
moreover from $\langle A_{P'} \#* V \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot V)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* V \rangle \langle (p \cdot A_P) \#* V \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* V$ **by** *simp*
moreover from $\langle A_{P'} \#* W \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot W)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* W \rangle \langle (p \cdot A_P) \#* W \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* W$ **by** *simp*
moreover from $\langle A_{P'} \#* X \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot X)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* X \rangle \langle (p \cdot A_P) \#* X \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* X$ **by** *simp*
moreover from $\langle A_{P'} \#* Y \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot Y)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* Y \rangle \langle (p \cdot A_P) \#* Y \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* Y$ **by** *simp*
moreover from $\langle A_{P'} \#* Z \rangle$ **have** $(p \cdot A_{P'}) \#* (p \cdot Z)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* Z \rangle \langle (p \cdot A_P) \#* Z \rangle$ *Sp* **have** $(p \cdot A_{P'}) \#* Z$ **by** *simp*
moreover from $\langle (bn(q \cdot \alpha)) \#* V \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot V)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* V \rangle \langle (p \cdot A_P) \#* V \rangle$ *Sp* **have** $bn((p \cdot q) \cdot \alpha) \#* V$
by(*simp add: perm-compose eqvts*)

moreover from $\langle (bn(q \cdot \alpha)) \#* W \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot W)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* W \rangle \langle (p \cdot A_P) \#* W \rangle$ **Sp have** $bn((p \cdot q) \cdot \alpha) \#* W$
by(*simp add: perm-compose eqts*)
moreover from $\langle (bn(q \cdot \alpha)) \#* X \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot X)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* X \rangle \langle (p \cdot A_P) \#* X \rangle$ **Sp have** $bn((p \cdot q) \cdot \alpha) \#* X$
by(*simp add: perm-compose eqts*)
moreover from $\langle (bn(q \cdot \alpha)) \#* Y \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot Y)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* Y \rangle \langle (p \cdot A_P) \#* Y \rangle$ **Sp have** $bn((p \cdot q) \cdot \alpha) \#* Y$
by(*simp add: perm-compose eqts*)
moreover from $\langle (bn(q \cdot \alpha)) \#* Z \rangle$ **have** $(p \cdot bn(q \cdot \alpha)) \#* (p \cdot Z)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_P \#* \alpha \rangle \langle (p \cdot A_P) \#* \alpha \rangle \langle A_P \#* Z \rangle \langle (p \cdot A_P) \#* Z \rangle$ **Sp have** $bn((p \cdot q) \cdot \alpha) \#* Z$
by(*simp add: perm-compose eqts*)
moreover from $\langle distinct A_P' \rangle$ **have** $distinct(p \cdot A_P')$ **by** *simp*
ultimately show *?case*
by(*elim cAlpha*)
next
case(*cInput* $\Psi M K xvec N Tvec P C C' \alpha P' V W X Y Z$)
moreover obtain $A_P \Psi_P$ **where** $extractFrame(P[xvec::=Tvec]) = \langle A_P, \Psi_P \rangle$
and $distinct A_P$
and $A_P \#* (C, P[xvec::=Tvec], \alpha, P', V, W, X, Y, Z, N)$
by(*rule freshFrame*)
moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(*blast intro: Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case*
by(*intro cInput*) (*assumption* | *simp add: residualInject*)+
next
case(*cBrInput* $\Psi M K xvec N Tvec P C C' \alpha P' V W X Y Z$)
moreover obtain $A_P \Psi_P$ **where** $extractFrame(P[xvec::=Tvec]) = \langle A_P, \Psi_P \rangle$
and $distinct A_P$
and $A_P \#* (C, P[xvec::=Tvec], \alpha, P', V, W, X, Y, Z, N)$
by(*rule freshFrame*)
moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(*blast intro: Identity Commutativity AssertionStatEqTrans*)
ultimately show *?case*
by(*intro cBrInput*) (*assumption* | *simp add: residualInject*)+
next
case(*cOutput* $\Psi M K N P C C' \alpha P' V W X Y Z$)
moreover obtain $A_P \Psi_P$ **where** $extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $distinct$
 A_P
and $A_P \#* (C, C', P, \alpha, N, P', V, W, X, Y, Z)$
by(*rule freshFrame*)

moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(blast intro: Identity Commutativity AssertionStatEqTrans)
ultimately show ?case **by**(simp add: residualInject)
next
case(cBrOutput Ψ M K N P C C' α P' V W X Y Z)
moreover obtain A_P Ψ_P **where** $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** *distinct*
 A_P
and $A_P \#* (C, C', P, \alpha, N, P', V, W, X, Y, Z)$
by(rule freshFrame)
moreover have $\mathbf{1} \otimes \Psi_P \simeq \Psi_P$
by(blast intro: Identity Commutativity AssertionStatEqTrans)
ultimately show ?case **by**(simp add: residualInject)
next
case(cCase Ψ P φ Cs A_P Ψ_P C C' α P' V W X Y Z)
moreover from $\langle \text{bn } \alpha \#* (Cases\ Cs) \rangle$ $\langle (\varphi, P) \in \text{set } Cs \rangle$ **have** $\text{bn } \alpha \#* P$
by(auto dest: memFreshChain)
ultimately obtain p Ψ' $A_{P'}$ $\Psi_{P'}$ **where** $S: \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha))$
 α)
and $\text{Fr}P'$: $\text{extractFrame } P' = \langle A_{P'}, \Psi_{P'} \rangle$
and $\text{Peg}P'$: $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$ **and** *distinct* $A_{P'}$
and $A_{P'} \#* C$ **and** $A_{P'} \#* P'$ **and** $A_{P'} \#* \alpha$ **and** $A_{P'} \#* (p \cdot \alpha)$
and $A_{P'} \#* V$ **and** $A_{P'} \#* W$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* Z$
and *distinctPerm* p **and** $(\text{bn}(p \cdot \alpha)) \#* \alpha$ **and** $(\text{bn}(p \cdot \alpha)) \#* P'$
and $(\text{bn}(p \cdot \alpha)) \#* C'$ **and** $(\text{bn}(p \cdot \alpha)) \#* V$ **and** $(\text{bn}(p \cdot \alpha)) \#* W$ **and**
 $(\text{bn}(p \cdot \alpha)) \#* X$ **and** $(\text{bn}(p \cdot \alpha)) \#* Y$ **and** $(\text{bn}(p \cdot \alpha)) \#* Z$
apply –
apply (rule cCase)
apply (assumption | simp (no-asm-use))+
done
moreover from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $(p \cdot \Psi_P) \simeq (p \cdot \mathbf{1})$
by(simp add: AssertionStatEqClosed)
then have $(p \cdot \Psi_P) \simeq \mathbf{1}$ **by**(simp add: permBottom)
with $\text{Peg}P'$ **have** $(\mathbf{1} \otimes \Psi') \simeq \Psi_{P'}$
by(metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym)
ultimately show ?case **using** cCase $\langle \text{bn } \alpha \#* P \rangle$
by(intro cCase(22)) (assumption | simp)+
next
case(cPar1 Ψ Ψ_Q P α P' A_Q Q A_P Ψ_P C C' α' PQ' V W X Y Z)
have $\text{Fr}P$: $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** $\text{Fr}Q$: $\text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle$
by fact+

note $\langle \text{bn } \alpha' \#* \text{subject } \alpha' \rangle$
moreover from $\langle \text{bn } \alpha' \#* (P \parallel Q) \rangle$ **have** $\text{bn } \alpha' \#* P$ **and** $\text{bn } \alpha' \#* Q$ **by**
simp+
moreover with $\text{Fr}P$ $\text{Fr}Q$ $\langle A_P \#* \alpha' \rangle$ $\langle A_Q \#* \alpha' \rangle$ **have** $\text{bn } \alpha' \#* \Psi_P$ **and** bn
 $\alpha' \#* \Psi_Q$
by(force dest: extractFrameFreshChain)+

moreover note $\langle bn \ \alpha' \ \#* \ V \rangle \langle bn \ \alpha' \ \#* \ W \rangle$
moreover from $\langle bn \ \alpha' \ \#* \ X \rangle \langle A_Q \ \#* \ \alpha' \rangle$ **have** $bn \ \alpha' \ \#* \ (X @ A_Q)$ **by simp**
moreover from $\langle bn \ \alpha' \ \#* \ Y \rangle \langle bn \ \alpha' \ \#* \ \Psi_Q \rangle$ **have** $bn \ \alpha' \ \#* \ (\Psi_Q \# Y)$ **by simp**
moreover from $\langle bn \ \alpha' \ \#* \ Z \rangle \langle bn \ \alpha' \ \#* \ Q \rangle$ **have** $bn \ \alpha' \ \#* \ (Q \# Z)$ **by simp**
moreover note $\langle A_P \ \#* \ V \rangle \langle A_P \ \#* \ W \rangle$
moreover from $\langle A_P \ \#* \ X \rangle \langle A_P \ \#* \ A_Q \rangle$ **have** $A_P \ \#* \ (X @ A_Q)$ **by simp**
moreover from $\langle A_P \ \#* \ Y \rangle \langle A_P \ \#* \ \Psi_Q \rangle$ **have** $A_P \ \#* \ (\Psi_Q \# Y)$ **by force**
moreover from $\langle A_P \ \#* \ Z \rangle \langle A_P \ \#* \ Q \rangle$ **have** $A_P \ \#* \ (Q \# Z)$ **by simp**
moreover from $\langle \alpha \prec (P' \parallel Q) = \alpha' \prec PQ' \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle bn \ \alpha' \ \#* \ Q \rangle \langle bn \ \alpha \ \#* \ \alpha' \rangle$
obtain P'' **where** $A: \alpha \prec P' = \alpha' \prec P''$ **and** $PQ' = P'' \parallel Q$
by(metis actionPar1Dest')
moreover from $\langle A_P \ \#* \ PQ' \rangle \langle PQ' = P'' \parallel Q \rangle$ **have** $A_P \ \#* \ P''$ **by simp**
ultimately obtain $p \ \Psi' \ \Psi_{P'} \ A_{P'}$ **where** $S: set \ p \subseteq set(bn \ \alpha') \times set(bn(p \cdot \alpha'))$ **and** $PeqP': ((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_{P'}$
and $distinctPerm \ p$ **and** $(bn(p \cdot \alpha')) \ \#* \ C'$ **and** $FrP': extractFrame \ P'' = \langle A_{P'}, \Psi_{P'} \rangle$
and $A_{P'} \ \#* \ P''$ **and** $A_{P'} \ \#* \ \alpha' \ A_{P'} \ \#* \ (p \cdot \alpha')$ **and** $A_{P'} \ \#* \ C$
and $(bn(p \cdot \alpha')) \ \#* \ \alpha'$ **and** $(bn(p \cdot \alpha')) \ \#* \ P''$ **and** $distinct \ A_{P'}$
and $A_{P'} \ \#* \ V$ **and** $A_{P'} \ \#* \ W$ **and** $A_{P'} \ \#* \ (X @ A_Q)$ **and** $A_{P'} \ \#* \ (\Psi_Q \# Y)$
and $A_{P'} \ \#* \ (Q \# Z)$ **and** $(bn(p \cdot \alpha')) \ \#* \ V$ **and** $(bn(p \cdot \alpha')) \ \#* \ W$ **and** $(bn(p \cdot \alpha')) \ \#* \ (X @ A_Q)$ **and** $(bn(p \cdot \alpha')) \ \#* \ (\Psi_Q \# Y)$
and $(bn(p \cdot \alpha')) \ \#* \ (Q \# Z)$ **using** $cPar1$
by(elim cPar1)

then have $A_{P'} \ \#* \ Q$ **and** $A_{P'} \ \#* \ Z$ **and** $A_{P'} \ \#* \ A_Q$ **and** $A_{P'} \ \#* \ X$ **and** $A_{P'} \ \#* \ \Psi_Q$ **and** $A_{P'} \ \#* \ Y$
and $(bn(p \cdot \alpha')) \ \#* \ A_Q$ **and** $(bn(p \cdot \alpha')) \ \#* \ X$ **and** $(bn(p \cdot \alpha')) \ \#* \ Y$ **and** $(bn(p \cdot \alpha')) \ \#* \ Z$ **and** $(bn(p \cdot \alpha')) \ \#* \ \Psi_Q$
and $(bn(p \cdot \alpha')) \ \#* \ Q$
by(simp del: freshChainSimps)+

from $\langle A_Q \ \#* \ PQ' \rangle \langle PQ' = P'' \parallel Q \rangle \langle A_{P'} \ \#* \ A_Q \rangle \langle FrP' \rangle$ **have** $A_Q \ \#* \ \Psi_{P'}$
by(force dest: extractFrameFreshChain)

note S
moreover from $PeqP'$ **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq \Psi_{P'} \otimes (p \cdot \Psi_Q)$
by(simp add: eqts) (metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity)

with $\langle (bn(p \cdot \alpha')) \ \#* \ \Psi_Q \rangle \langle bn \ \alpha' \ \#* \ \Psi_Q \rangle \langle S \rangle$ **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq \Psi_{P'} \otimes \Psi_Q$
by simp

moreover from $\langle PQ' = P'' \parallel Q \rangle \langle A_{P'} \ \#* \ A_Q \rangle \langle A_{P'} \ \#* \ \Psi_Q \rangle \langle A_Q \ \#* \ \Psi_{P'} \rangle \langle A_Q \ \#* \ PQ' \rangle \langle FrP' \rangle \langle FrQ \rangle$ **have** $extractFrame \ PQ' = \langle (A_{P'} @ A_Q), \Psi_{P'} \otimes \Psi_Q \rangle$
by simp

moreover note $\langle distinctPerm \ p \rangle \langle (bn(p \cdot \alpha')) \ \#* \ C' \rangle$
moreover from $\langle A_{P'} \ \#* \ P'' \rangle \langle A_{P'} \ \#* \ Q \rangle \langle PQ' = P'' \parallel Q \rangle$ **have** $A_{P'} \ \#* \ PQ'$
by simp

moreover note $\langle A_Q \ \#* \ PQ' \rangle \langle A_{P'} \ \#* \ \alpha' \rangle \langle A_Q \ \#* \ \alpha' \rangle \langle A_{P'} \ \#* \ C \rangle \langle A_Q \ \#* \ C \rangle$

$\langle (bn(p \cdot \alpha')) \#* \alpha' \rangle$
moreover from $\langle bn \alpha' \#* Q \rangle$ **have** $(bn(p \cdot \alpha')) \#* (p \cdot Q)$
by (*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*] *bnEqvt*[*symmetric*])
with $\langle bn \alpha \#* Q \rangle \langle (bn(p \cdot \alpha')) \#* Q \rangle$ **have** $(bn(p \cdot \alpha')) \#* Q$ **by** *simp*
with $\langle (bn(p \cdot \alpha')) \#* P'' \rangle \langle PQ' = P'' \parallel Q \rangle$ **have** $(bn(p \cdot \alpha')) \#* PQ'$ **by** *simp*
moreover from $\langle A_{P'} \#* \alpha' \rangle \langle A_Q \#* \alpha' \rangle$ **have** $(A_{P'} @ A_Q) \#* \alpha'$ **by** *simp*
moreover from $\langle A_{P'} \#* C \rangle \langle A_Q \#* C \rangle$ **have** $(A_{P'} @ A_Q) \#* C$ **by** *simp*
moreover from $\langle A_{P'} \#* V \rangle \langle A_Q \#* V \rangle$ **have** $(A_{P'} @ A_Q) \#* V$ **by** *simp*
moreover from $\langle A_{P'} \#* W \rangle \langle A_Q \#* W \rangle$ **have** $(A_{P'} @ A_Q) \#* W$ **by** *simp*
moreover from $\langle A_{P'} \#* X \rangle \langle A_Q \#* X \rangle$ **have** $(A_{P'} @ A_Q) \#* X$ **by** *simp*
moreover from $\langle A_{P'} \#* Y \rangle \langle A_Q \#* Y \rangle$ **have** $(A_{P'} @ A_Q) \#* Y$ **by** *simp*
moreover from $\langle A_{P'} \#* Z \rangle \langle A_Q \#* Z \rangle$ **have** $(A_{P'} @ A_Q) \#* Z$ **by** *simp*
moreover from $\langle A_{P'} \#* PQ' \rangle \langle A_Q \#* PQ' \rangle$ **have** $(A_{P'} @ A_Q) \#* PQ'$ **by** *simp*
moreover from $\langle A_{P'} \#* \alpha' \rangle \langle A_Q \#* \alpha' \rangle$ **have** $(A_{P'} @ A_Q) \#* \alpha'$ **by** *simp*
moreover from $\langle A_Q \#* \alpha' \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \alpha')$
by (*simp only: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*] *bnEqvt*[*symmetric*])
with $S \langle A_Q \#* \alpha' \rangle \langle (bn(p \cdot \alpha')) \#* A_Q \rangle$ **have** $A_Q \#* (p \cdot \alpha')$
by *simp*
with $\langle A_{P'} \#* (p \cdot \alpha') \rangle \langle A_Q \#* \alpha' \rangle \langle (bn(p \cdot \alpha')) \#* A_Q \rangle$ **have** $(A_{P'} @ A_Q) \#*$
 $(p \cdot \alpha')$
by *simp*
moreover from $\langle A_{P'} \#* A_Q \rangle \langle distinct A_{P'} \rangle \langle distinct A_Q \rangle$ **have** $distinct(A_{P'} @ A_Q)$
by *auto*
moreover note $\langle (bn(p \cdot \alpha')) \#* V \rangle \langle (bn(p \cdot \alpha')) \#* W \rangle \langle (bn(p \cdot \alpha')) \#* X \rangle$
 $\langle (bn(p \cdot \alpha')) \#* Y \rangle \langle (bn(p \cdot \alpha')) \#* Z \rangle$
ultimately show *?case using cPar1*
by *metis*
next
case (*cPar2* $\Psi \Psi_P Q \alpha Q' A_P P A_Q \Psi_Q C C' \alpha' PQ' V W X Y Z$)
have $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $FrQ: extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle$
by *fact+*

note $\langle bn \alpha' \#* subject \alpha' \rangle$
moreover from $\langle bn \alpha' \#* (P \parallel Q) \rangle$ **have** $bn \alpha' \#* Q$ **and** $bn \alpha' \#* P$ **by**
simp+
moreover with $FrP FrQ \langle A_P \#* \alpha' \rangle \langle A_Q \#* \alpha' \rangle$ **have** $bn \alpha' \#* \Psi_P$ **and** bn
 $\alpha' \#* \Psi_Q$
by (*force dest: extractFrameFreshChain*)

moreover note $\langle bn \alpha' \#* V \rangle \langle bn \alpha' \#* W \rangle$
moreover from $\langle bn \alpha' \#* X \rangle \langle A_P \#* \alpha' \rangle$ **have** $bn \alpha' \#* (X @ A_P)$ **by** *simp*
moreover from $\langle bn \alpha' \#* Y \rangle \langle bn \alpha' \#* \Psi_P \rangle$ **have** $bn \alpha' \#* (\Psi_P \# Y)$ **by** *simp*
moreover from $\langle bn \alpha' \#* Z \rangle \langle bn \alpha' \#* P \rangle$ **have** $bn \alpha' \#* (P \# Z)$ **by** *simp*
moreover note $\langle A_Q \#* V \rangle \langle A_Q \#* W \rangle$
moreover from $\langle A_Q \#* X \rangle \langle A_P \#* A_Q \rangle$ **have** $A_Q \#* (X @ A_P)$ **by** *simp*
moreover from $\langle A_Q \#* Y \rangle \langle A_Q \#* \Psi_P \rangle$ **have** $A_Q \#* (\Psi_P \# Y)$ **by** *force*
moreover from $\langle A_Q \#* Z \rangle \langle A_Q \#* P \rangle$ **have** $A_Q \#* (P \# Z)$ **by** *simp*
moreover from $\langle \alpha \prec (P \parallel Q') = \alpha' \prec PQ' \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha' \#* P \rangle \langle bn$

$\alpha \#^* \alpha'$
obtain Q'' **where** $A: \alpha \prec Q' = \alpha' \prec Q''$ **and** $PQ' = P \parallel Q''$
by(*metis actionPar2Dest*)
moreover from $\langle A_Q \#^* PQ' \rangle \langle PQ' = P \parallel Q'' \rangle$ **have** $A_Q \#^* Q''$ **by** *simp*
ultimately obtain $p \Psi' A_Q' \Psi_Q'$ **where** $S: \text{set } p \subseteq \text{set}(bn \alpha') \times \text{set}(bn(p \cdot \alpha'))$ **and** $QeqQ': ((p \cdot \Psi_Q) \otimes \Psi') \simeq \Psi_Q'$
and *distinctPerm* p **and** $(bn(p \cdot \alpha')) \#^* C'$ **and** $FrQ': \text{extractFrame } Q'' = \langle A_Q', \Psi_Q' \rangle$
and $A_Q' \#^* Q''$ **and** $A_Q' \#^* \alpha' A_Q' \#^* (p \cdot \alpha')$ **and** $A_Q' \#^* C$
and $(bn(p \cdot \alpha')) \#^* \alpha'$ **and** $(bn(p \cdot \alpha')) \#^* Q''$ **and** *distinct* A_Q'
and $A_Q' \#^* V$ **and** $A_Q' \#^* W$ **and** $A_Q' \#^* (X @ A_P)$ **and** $A_Q' \#^* (\Psi_P \# Y)$
and $A_Q' \#^* (P \# Z)$ **and** $(bn(p \cdot \alpha')) \#^* V$ **and** $(bn(p \cdot \alpha')) \#^* W$ **and** $(bn(p \cdot \alpha')) \#^* (X @ A_P)$ **and** $(bn(p \cdot \alpha')) \#^* (\Psi_P \# Y)$
and $(bn(p \cdot \alpha')) \#^* (P \# Z)$ **using** *cPar2*
by(*elim cPar2*)

then have $A_Q' \#^* P$ **and** $A_Q' \#^* Z$ **and** $A_Q' \#^* A_P$ **and** $A_Q' \#^* X$ **and** $A_Q' \#^* \Psi_P$ **and** $A_Q' \#^* Y$
and $(bn(p \cdot \alpha')) \#^* A_P$ **and** $(bn(p \cdot \alpha')) \#^* X$ **and** $(bn(p \cdot \alpha')) \#^* Y$ **and** $(bn(p \cdot \alpha')) \#^* Z$ **and** $(bn(p \cdot \alpha')) \#^* \Psi_P$
and $(bn(p \cdot \alpha')) \#^* P$
by(*simp del: freshChainSimps*)+

from $\langle A_P \#^* PQ' \rangle \langle PQ' = P \parallel Q'' \rangle \langle A_Q' \#^* A_P \rangle FrQ'$ **have** $A_P \#^* \Psi_Q'$
by(*force dest: extractFrameFreshChain*)
note S
moreover from $QeqQ'$ **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq (p \cdot \Psi_P) \otimes \Psi_Q'$
by(*simp add: eqts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)
with $\langle (bn(p \cdot \alpha')) \#^* \Psi_P \rangle \langle bn \alpha' \#^* \Psi_P \rangle S$ **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes \Psi') \simeq \Psi_P \otimes \Psi_Q'$
by *simp*
moreover from $\langle PQ' = P \parallel Q'' \rangle \langle A_Q' \#^* A_P \rangle \langle A_Q' \#^* \Psi_P \rangle \langle A_P \#^* \Psi_Q' \rangle \langle A_P \#^* PQ' \rangle FrQ' FrP$ **have** $\text{extractFrame } PQ' = \langle (A_P @ A_Q'), \Psi_P \otimes \Psi_Q' \rangle$
by *simp*
moreover note $\langle \text{distinctPerm } p \rangle \langle (bn(p \cdot \alpha')) \#^* C' \rangle$
moreover from $\langle A_Q' \#^* Q'' \rangle \langle A_Q' \#^* P \rangle \langle PQ' = P \parallel Q'' \rangle$ **have** $A_Q' \#^* PQ'$
by *simp*
moreover note $\langle A_Q \#^* PQ' \rangle \langle A_Q' \#^* \alpha' \rangle \langle A_Q \#^* \alpha' \rangle \langle A_Q' \#^* C \rangle \langle A_Q \#^* C \rangle \langle (bn(p \cdot \alpha')) \#^* \alpha' \rangle$
moreover from $\langle bn \alpha' \#^* Q \rangle$ **have** $(bn(p \cdot \alpha')) \#^* (p \cdot Q)$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst] bnEqvt[symmetric]*)
with $\langle bn \alpha \#^* P \rangle \langle (bn(p \cdot \alpha')) \#^* P \rangle S$ **have** $(bn(p \cdot \alpha')) \#^* P$ **by** *simp*
with $\langle (bn(p \cdot \alpha')) \#^* Q'' \rangle \langle PQ' = P \parallel Q'' \rangle$ **have** $(bn(p \cdot \alpha')) \#^* PQ'$ **by** *simp*
moreover from $\langle A_Q' \#^* \alpha' \rangle \langle A_P \#^* \alpha' \rangle$ **have** $(A_P @ A_Q') \#^* \alpha'$ **by** *simp*
moreover from $\langle A_Q' \#^* C \rangle \langle A_P \#^* C \rangle$ **have** $(A_P @ A_Q') \#^* C$ **by** *simp*
moreover from $\langle A_Q' \#^* V \rangle \langle A_P \#^* V \rangle$ **have** $(A_P @ A_Q') \#^* V$ **by** *simp*
moreover from $\langle A_Q' \#^* W \rangle \langle A_P \#^* W \rangle$ **have** $(A_P @ A_Q') \#^* W$ **by** *simp*
moreover from $\langle A_Q' \#^* X \rangle \langle A_P \#^* X \rangle$ **have** $(A_P @ A_Q') \#^* X$ **by** *simp*

```

moreover from  $\langle A_{Q'} \#* Y \rangle \langle A_P \#* Y \rangle$  have  $(A_P @ A_{Q'}) \#* Y$  by simp
moreover from  $\langle A_{Q'} \#* Z \rangle \langle A_P \#* Z \rangle$  have  $(A_P @ A_{Q'}) \#* Z$  by simp
moreover from  $\langle A_{Q'} \#* PQ' \rangle \langle A_P \#* PQ' \rangle$  have  $(A_P @ A_{Q'}) \#* PQ'$  by simp
moreover from  $\langle A_{Q'} \#* \alpha' \rangle \langle A_P \#* \alpha' \rangle$  have  $(A_P @ A_{Q'}) \#* \alpha'$  by simp
moreover from  $\langle A_P \#* \alpha' \rangle$  have  $(p \cdot A_P) \#* (p \cdot \alpha')$ 
by (simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst] bnEqut[symmetric])
with  $S \langle A_P \#* \alpha' \rangle \langle bn(p \cdot \alpha') \#* A_P \rangle$  have  $A_P \#* (p \cdot \alpha')$ 
  by simp
with  $\langle A_{Q'} \#* (p \cdot \alpha') \rangle \langle A_P \#* \alpha' \rangle \langle bn(p \cdot \alpha') \#* A_P \rangle S$  have  $(A_P @ A_{Q'}) \#*$ 
 $(p \cdot \alpha')$ 
  by simp
moreover note  $\langle (bn(p \cdot \alpha')) \#* V \rangle \langle (bn(p \cdot \alpha')) \#* W \rangle \langle (bn(p \cdot \alpha')) \#* X \rangle$ 
 $\langle (bn(p \cdot \alpha')) \#* Y \rangle \langle (bn(p \cdot \alpha')) \#* Z \rangle$ 
moreover from  $\langle A_{Q'} \#* A_P \rangle \langle distinct A_P \rangle \langle distinct A_{Q'} \rangle$  have  $distinct(A_P @ A_{Q'})$ 
by auto
  ultimately show ?case using cPar2
    by metis
next
  case cComm1
    then show ?case by (simp add: residualInject)
next
  case cComm2
    then show ?case by (simp add: residualInject)
next
  case (cBrMerge  $\Psi \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q C C' \alpha PQ' V W X Y Z$ )
    have FrP:  $extractFrame P = \langle A_P, \Psi_P \rangle$  and FrQ:  $extractFrame Q = \langle A_Q,$ 
 $\Psi_Q \rangle$ 
    by fact+

have  $bn (iM(N)) \#* Q$  by simp
have  $bn (iM(N)) \#* Q'$  by simp
have  $bn (iM(N)) \#* \alpha$  by simp
have  $bn (iM(N)) \#* P$  by simp
have  $bn (iM(N)) \#* P'$  by simp
have  $bn (iM(N)) \#* \alpha$  by simp

from  $\langle iM(N) \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ 
have empty $\alpha$ :  $bn \alpha = (\langle \rangle :: name\ list)$  and
 $\alpha = (iM(N))$ 
  by (simp add: residualInject)+
then have  $bn \alpha \#* Q$  and  $bn \alpha \#* Q'$ 
and  $bn \alpha \#* P$  and  $bn \alpha \#* P'$  by simp+

from  $\langle bn \alpha = \langle \rangle \rangle$ 
have  $bn \alpha \#* subject \alpha$  and  $bn \alpha \#* P$  and  $bn \alpha \#* Q$ 
and  $bn \alpha \#* \Psi_P$  and  $bn \alpha \#* \Psi_Q$  and  $bn \alpha \#* C'$ 
and  $bn \alpha \#* (X @ A_Q)$ 
and  $bn \alpha \#* (\Psi_Q \# Y)$  and  $bn \alpha \#* (Q \# Z)$  by simp+

```

moreover note $\langle A_P \#* P \rangle \langle A_P \#* \alpha \rangle \langle A_P \#* C \rangle \langle A_P \#* C' \rangle$
 $\langle \alpha \neq \tau \rangle$

moreover from $\langle A_P \#* X \rangle \langle A_P \#* A_Q \rangle$ **have** $A_P \#* (X @ A_Q)$ **by simp**
moreover from $\langle A_P \#* Y \rangle \langle A_P \#* \Psi_Q \rangle$ **have** $A_P \#* (\Psi_Q \# Y)$ **by force**
moreover from $\langle A_P \#* Z \rangle \langle A_P \#* Q \rangle$ **have** $A_P \#* (Q \# Z)$ **by simp**
moreover from $\langle iM(\downarrow N) \prec (P' \parallel Q') = \alpha \prec PQ' \rangle \langle bn(iM(\downarrow N)) \#* Q' \rangle \langle bn$
 $\alpha \#* Q' \rangle \langle bn(iM(\downarrow N)) \#* \alpha \rangle$
obtain P'' **where** $A: iM(\downarrow N) \prec P' = \alpha \prec P''$ **and** $PQ' = P'' \parallel Q'$
by(metis actionPar1Dest')
moreover from $\langle A_P \#* PQ' \rangle \langle PQ' = P'' \parallel Q' \rangle$ **have** $A_P \#* P''$ **by simp**
ultimately obtain $p P \Psi' A_P' \Psi_{P'}$ **where** $Sp: set p \subseteq set(bn \alpha) \times set(bn(p$
 $\cdot \alpha))$ **and** $PeqP': ((p \cdot \Psi_P) \otimes P \Psi') \simeq \Psi_{P'}$
and $distinctPerm p$ **and** $FrP': extractFrame P'' = \langle A_P', \Psi_{P'} \rangle$
and $A_P' \#* P''$ **and** $A_P' \#* \alpha$ $A_P' \#* (p \cdot \alpha)$ **and** $A_P' \#* C$ **and** $(bn(p \cdot$
 $\alpha)) \#* C'$
and $(bn(p \cdot \alpha)) \#* \alpha$ **and** $(bn(p \cdot \alpha)) \#* P''$ **and** $distinct A_P'$
and $A_P' \#* V$ **and** $A_P' \#* W$ **and** $A_P' \#* (X @ A_Q)$ **and** $A_P' \#* (\Psi_Q \# Y)$
and $A_P' \#* (Q \# Z)$ **and** $(bn(p \cdot \alpha)) \#* V$ **and** $(bn(p \cdot \alpha)) \#* W$ **and** $(bn(p$
 $\cdot \alpha)) \#* (X @ A_Q)$ **and** $(bn(p \cdot \alpha)) \#* (\Psi_Q \# Y)$
and $(bn(p \cdot \alpha)) \#* (Q \# Z)$ **using** $cBrMerge$
by(elim cBrMerge)

then have $A_P' \#* Q$ **and** $A_P' \#* Z$ **and** $A_P' \#* A_Q$ **and** $A_P' \#* X$ **and** A_P'
 $\#* \Psi_Q$ **and** $A_P' \#* Y$
and $(bn(p \cdot \alpha)) \#* A_Q$ **and** $(bn(p \cdot \alpha)) \#* X$ **and** $(bn(p \cdot \alpha)) \#* Y$ **and**
 $(bn(p \cdot \alpha)) \#* Z$ **and** $(bn(p \cdot \alpha)) \#* \Psi_Q$
and $(bn(p \cdot \alpha)) \#* Q$
by(simp del: freshChainSimps)+

from $\langle A_Q \#* PQ' \rangle \langle PQ' = (P'' \parallel Q') \rangle$ **have** $A_Q \#* P''$ **by simp**
with $\langle extractFrame P'' = \langle A_P', \Psi_{P'} \rangle \rangle \langle A_P' \#* A_Q \rangle$ **have** $A_Q \#* \Psi_{P'}$
by (metis extractFrameFreshChain freshFrameDest)

from $\langle bn \alpha = [] \rangle$
have $bn \alpha \#* subject \alpha$ **and** $bn \alpha \#* Q$ **and** $bn \alpha \#* P''$
and $bn \alpha \#* \Psi_Q$ **and** $bn \alpha \#* \Psi_{P'}$ **and** $bn \alpha \#* C'$
and $bn \alpha \#* (X @ A_P')$
and $bn \alpha \#* (\Psi_{P'} \# Y)$ **and** $bn \alpha \#* (P'' \# Z)$ **by simp**+

moreover note $\langle A_P' \#* Q \rangle \langle A_Q \#* \alpha \rangle \langle A_Q \#* C \rangle \langle A_Q \#* C' \rangle$
 $\langle \alpha \neq \tau \rangle$

moreover from $\langle A_Q \#* X \rangle \langle A_P' \#* A_Q \rangle$ **have** $A_Q \#* (X @ A_P')$ **by simp**
moreover from $\langle A_Q \#* Y \rangle \langle A_Q \#* \Psi_{P'} \rangle$ **have** $A_Q \#* (\Psi_{P'} \# Y)$ **by force**
moreover from $\langle A_Q \#* Z \rangle \langle A_Q \#* P'' \rangle$ **have** $A_Q \#* (P'' \# Z)$ **by simp**
moreover from $\langle iM(\downarrow N) \prec (P' \parallel Q') = \alpha \prec PQ' \rangle \langle bn(iM(\downarrow N)) \#* P' \rangle \langle bn$
 $\alpha \#* P' \rangle \langle bn(iM(\downarrow N)) \#* \alpha \rangle$
obtain Q'' **where** $A: iM(\downarrow N) \prec Q' = \alpha \prec Q''$ **and** $PQ' = P' \parallel Q''$

by(*metis actionPar2Dest*)

moreover from $\langle PQ' = P'' \parallel Q' \rangle$ $\langle PQ' = P' \parallel Q'' \rangle$
have $PQ' = P'' \parallel Q''$
by (*simp add: psi.inject*)

moreover from $\langle A_Q \#* PQ' \rangle$ $\langle PQ' = P'' \parallel Q'' \rangle$ **have** $A_Q \#* Q''$ **by** *simp*
ultimately obtain $q \Psi' A_Q' \Psi_Q'$ **where** Sq : *set* $q \subseteq \text{set}(bn \alpha) \times \text{set}(bn(q \cdot \alpha))$ **and** $QeqQ'$: $((q \cdot \Psi_Q) \otimes \Psi') \simeq \Psi_Q'$
and *distinctPerm* q **and** FrQ' : *extractFrame* $Q'' = \langle A_Q', \Psi_Q' \rangle$
and $A_Q' \#* Q''$ **and** $A_Q' \#* \alpha A_Q' \#* (q \cdot \alpha)$ **and** $A_Q' \#* C$ **and** $(bn(q \cdot \alpha)) \#* C'$
and $(bn(q \cdot \alpha)) \#* \alpha$ **and** $(bn(q \cdot \alpha)) \#* Q''$ **and** *distinct* A_Q'
and $A_Q' \#* V$ **and** $A_Q' \#* W$ **and** $A_Q' \#* (X @ A_P')$ **and** $A_Q' \#* (\Psi_P' \# Y)$
and $A_Q' \#* (P'' \# Z)$ **and** $(bn(q \cdot \alpha)) \#* V$ **and** $(bn(q \cdot \alpha)) \#* W$ **and** $(bn(q \cdot \alpha)) \#* (X @ A_P')$ **and** $(bn(q \cdot \alpha)) \#* (\Psi_P' \# Y)$
and $(bn(q \cdot \alpha)) \#* (P'' \# Z)$ **using** *cBrMerge*
by(*elim cBrMerge(6)[where bb= α]*) (*rule refl | assumption*)+

then have $A_Q' \#* P''$ **and** $A_Q' \#* Z$ **and** $A_Q' \#* A_P'$ **and** $A_Q' \#* X$ **and**
 $A_Q' \#* \Psi_P'$ **and** $A_Q' \#* Y$
and $(bn(q \cdot \alpha)) \#* A_P'$ **and** $(bn(q \cdot \alpha)) \#* X$ **and** $(bn(q \cdot \alpha)) \#* Y$ **and**
 $(bn(q \cdot \alpha)) \#* Z$ **and** $(bn(q \cdot \alpha)) \#* \Psi_P'$
and $(bn(q \cdot \alpha)) \#* P''$
by(*simp del: freshChainSimps*)+

from $Sp Sq \langle bn \alpha = [] \rangle$ **have** $p = ([::\text{name prm}])$ **and** $q = ([::\text{name prm}])$
and $p = q$
by *simp*+

from $\langle A_Q' \#* P'' \rangle$ $\langle A_Q' \#* A_P' \rangle$ $\langle \text{extractFrame } P'' = \langle A_P', \Psi_P' \rangle \rangle$ **have** $A_Q' \#* \Psi_P'$
by (*metis extractFrameFreshChain freshFrameDest*)

from $\langle A_P' \#* \alpha \rangle$ $\langle \alpha = \iota M(N) \rangle$ **have** $A_P' \#* M$ **and** $A_P' \#* N$
by *simp*+

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q' \rangle$ $\langle A_P' \#* Q \rangle$ $\langle A_P' \#* N \rangle$
have $A_P' \#* Q'$
by(*simp add: brinputFreshChainDerivative*)

with $\langle PQ' = (P'' \parallel Q') \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$ **have** $A_P' \#* Q''$
by (*simp add: psi.inject*)

with $FrQ' \langle A_Q' \#* A_P' \rangle$ **have** $A_P' \#* \Psi_Q'$
by(*metis extractFrameFreshChain freshFrameDest*)

from $\langle A_P' \#* P'' \rangle$ $\langle A_P' \#* Q'' \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$
have $A_P' \#* PQ'$ **by** *simp*

from $\langle A_Q' \#* P'' \rangle$ $\langle A_Q' \#* Q'' \rangle$ $\langle PQ' = (P'' \parallel Q'') \rangle$
have $A_Q' \#* PQ'$ **by** *simp*

from $\text{Peq}P'$ **have** $((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes P\Psi') \simeq \Psi_{P'} \otimes (p \cdot \Psi_Q)$
by(*simp add: eqts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)

with $\langle p = [] \rangle$ **have** $((\Psi_P \otimes \Psi_Q) \otimes P\Psi') \simeq (\Psi_{P'} \otimes \Psi_Q)$ **by** *simp*
with $\text{Qeq}Q'$ **have** $((q \cdot (\Psi_Q \otimes \Psi_{P'})) \otimes \Psi') \simeq \Psi_{Q'} \otimes (q \cdot \Psi_{P'})$
by(*simp add: eqts*) (*metis Composition Associativity AssertionStatEqTrans AssertionStatEqSym Commutativity*)

with $\text{Peq}P'$ $\langle p = [] \rangle$ $\langle q = [] \rangle$ **have** $((q \cdot (\Psi_Q \otimes ((p \cdot \Psi_P) \otimes P\Psi')))) \otimes \Psi' \simeq \Psi_{Q'} \otimes (q \cdot \Psi_{P'})$
by(*simp add: eqts*) (*metis AssertionStatEqTrans Composition composition-Sym*)

with $\langle p = [] \rangle$ **have** $((q \cdot (\Psi_Q \otimes (\Psi_P \otimes P\Psi')))) \otimes \Psi' \simeq \Psi_{Q'} \otimes (q \cdot \Psi_{P'})$ **by** *simp*

with $\langle q = [] \rangle$ **have** $(q \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes \Psi') \simeq \Psi_{P'} \otimes \Psi_{Q'}$
by(*simp add: eqts*) (*metis AssertionStatEqTrans Commutativity Composition associativitySym*)

moreover note $\langle \text{set } q \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn } (q \cdot \alpha)) \rangle$

moreover from $\langle PQ' = P'' \parallel Q'' \rangle$ $\langle A_{Q'} \#* A_{P'} \rangle$ $\langle A_{P'} \#* \Psi_{Q'} \rangle$ $\langle A_{Q'} \#* \Psi_{P'} \rangle$
 $\langle A_{Q'} \#* PQ' \rangle$ $\text{Fr}P'$ $\text{Fr}Q'$ **have** $\text{extractFrame } PQ' = \langle (A_{P'} @ A_{Q'}), \Psi_{P'} \otimes \Psi_{Q'} \rangle$
by *simp*

moreover note $\langle \text{distinctPerm } q \rangle$

moreover from $\langle A_{P'} \#* PQ' \rangle$ $\langle A_{Q'} \#* PQ' \rangle$
have $(A_{P'} @ A_{Q'}) \#* PQ'$ **by** *simp*
moreover from $\langle A_{P'} \#* \alpha \rangle$ $\langle A_{Q'} \#* \alpha \rangle$
have $(A_{P'} @ A_{Q'}) \#* \alpha$ **by** *simp*
moreover with $\langle q = [] \rangle$
have $(A_{P'} @ A_{Q'}) \#* (q \cdot \alpha)$ **by** *simp*
moreover from $\langle A_{P'} \#* C \rangle$ $\langle A_{Q'} \#* C \rangle$
have $(A_{P'} @ A_{Q'}) \#* C$ **by** *simp*
moreover note $\langle \text{bn } (q \cdot \alpha) \#* C' \rangle$ $\langle \text{bn } (q \cdot \alpha) \#* \alpha \rangle$
moreover from $\langle \text{bn } \alpha = [] \rangle$
have $\text{bn } (q \cdot \alpha) \#* PQ'$

by (*metis Nominal.nil-eqvt bnEqvt freshSets*)

moreover from $\langle A_{P'} \#* V \rangle$ $\langle A_{Q'} \#* V \rangle$
have $(A_{P'} @ A_{Q'}) \#* V$ **by** *simp*
moreover from $\langle A_{P'} \#* W \rangle$ $\langle A_{Q'} \#* W \rangle$
have $(A_{P'} @ A_{Q'}) \#* W$ **by** *simp*
moreover from $\langle A_{P'} \#* X \rangle$ $\langle A_{Q'} \#* X \rangle$
have $(A_{P'} @ A_{Q'}) \#* X$ **by** *simp*
moreover from $\langle A_{P'} \#* Y \rangle$ $\langle A_{Q'} \#* Y \rangle$
have $(A_{P'} @ A_{Q'}) \#* Y$ **by** *simp*
moreover from $\langle A_{P'} \#* Z \rangle$ $\langle A_{Q'} \#* Z \rangle$
have $(A_{P'} @ A_{Q'}) \#* Z$ **by** *simp*

moreover from $\langle A_{Q'} \#* A_{P'} \rangle$ $\langle \text{distinct } A_{P'} \rangle$ $\langle \text{distinct } A_{Q'} \rangle$ **have** *dis-*

$\text{tinct}(A_P' @ A_Q')$ by *simp*
moreover note $\langle (bn(q \cdot \alpha)) \#* V \rangle \langle (bn(q \cdot \alpha)) \#* W \rangle \langle (bn(q \cdot \alpha)) \#* X \rangle$
 $\langle (bn(q \cdot \alpha)) \#* Y \rangle \langle (bn(q \cdot \alpha)) \#* Z \rangle$
ultimately show *?case*
by (*intro cBrMerge(47)*)
next
case (*cBrComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q \text{ xvec } Q' A_Q C C' \alpha PQ' V W$
 $X Y Z$)
have *FrP*: $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** *FrQ*: $\text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle$
by *fact+*

from $\langle \text{xvec} \#* \alpha \rangle$ **have** $\text{xvec} \#* bn \alpha$ **by** *simp*

from $\langle \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ **have** $\alpha \prec PQ' = \text{!}M(\nu * \text{xvec}) \langle N \rangle$
 $\prec P' \parallel Q'$ **by** *simp*
with $\langle \text{xvec} \#* (bn \alpha) \rangle$
obtain $Q'' r$ **where** *rPerm*: $\text{set } r \subseteq \text{set } (bn \alpha) \times \text{set } \text{xvec}$
and $PQ' = (r \cdot P') \parallel Q''$ **and** $\alpha \prec Q'' = \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec Q'$
by (*elim actionPar2Dest*) (*assumption* | *simp*) +
then have $\text{!}M(\nu * \text{xvec}) \langle N \rangle \prec Q' = \alpha \prec Q''$ **by** *simp*

from $\langle A_Q \#* PQ' \rangle \langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_Q \#* Q''$
by *simp*

from $\langle A_P \#* PQ' \rangle \langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_P \#* Q''$
by *simp*

from $\langle bn \alpha \#* (P \parallel Q) \rangle \langle A_P \#* \alpha \rangle$
have $bn \alpha \#* P$ **and** $A_P \#* bn \alpha$ **by** *simp+*
with $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ **have** $bn \alpha \#* \Psi_P$
by (*metis extractFrameFreshChain.freshFrameDest*)

from $\langle bn \alpha \#* (P \parallel Q) \rangle \langle A_Q \#* \alpha \rangle$
have $bn \alpha \#* Q$ **and** $A_Q \#* bn \alpha$ **by** *simp+*
with $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $bn \alpha \#* \Psi_Q$
by (*metis extractFrameFreshChain.freshFrameDest*)

from *rPerm* $\langle A_P \#* \text{xvec} \rangle \langle \text{xvec} \#* \Psi_P \rangle \langle A_P \#* bn \alpha \rangle \langle bn \alpha \#* \Psi_P \rangle$
have $r \cdot A_P = A_P$ **and** $r \cdot \Psi_P = \Psi_P$ **by** *simp+*

have $\text{!}M \langle N \rangle \prec P' = \text{!}M \langle N \rangle \prec P'$ **by** *simp*
moreover note $\langle A_P \#* P \rangle \langle A_P \#* C \rangle \langle A_P \#* C' \rangle$
moreover from $\langle A_P \#* M \rangle \langle A_P \#* N \rangle$ **have** $A_P \#* (\text{!}M \langle N \rangle)$ **by** *simp*
moreover note $\langle A_P \#* V \rangle$
moreover from $\langle A_P \#* W \rangle \langle A_P \#* \alpha \rangle$ **have** $A_P \#* (\alpha \# W)$ **by** *simp*
moreover from $\langle A_P \#* X \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \text{xvec} \rangle$ **have** $A_P \#* (X @ A_Q @ \text{xvec})$
by *simp*
moreover from $\langle A_P \#* Y \rangle \langle A_P \#* \Psi_Q \rangle$ **have** $A_P \#* (\Psi_Q \# Y)$ **by** *force*

moreover from $\langle A_P \#* Z \rangle \langle A_P \#* Q \rangle \langle A_P \#* Q' \rangle \langle A_P \#* Q'' \rangle$ **have** $A_P \#*$
 $(Q\#(Q'\#(Q''\#Z)))$ **by** *simp*

moreover note $\langle A_P \#* P' \rangle \langle A_P \#* Q \rangle$

ultimately obtain $p P\Psi' A_{P'} \Psi_{P'}$ **where** Sp : $set\ p \subseteq set(bn\ (\iota M(N))) \times$
 $set\ (bn(p \cdot (\iota M(N))))$ **and** $((p \cdot \Psi_P) \otimes P\Psi') \simeq \Psi_{P'}$

and *distinctPerm* p **and** *FrP'*: $extractFrame\ P' = \langle A_{P'}, \Psi_{P'} \rangle$

and $A_{P'} \#* P'$ **and** $A_{P'} \#* (\iota M(N))$ $A_{P'} \#* (p \cdot (\iota M(N)))$ **and** $A_{P'} \#* C$
and $(bn(p \cdot (\iota M(N)))) \#* C'$

and $(bn(p \cdot (\iota M(N)))) \#* (\iota M(N))$ **and** $(bn(p \cdot (\iota M(N)))) \#* P'$ **and**
distinct $A_{P'}$

and $A_{P'} \#* V$ **and** $A_{P'} \#* (\alpha\#W)$ **and** $A_{P'} \#* (X @ A_Q @ xvec)$ **and** $A_{P'} \#*$
 $(\Psi_Q\#Y)$

and $A_{P'} \#* (Q\#(Q'\#(Q''\#Z)))$ **and** $(bn(p \cdot (\iota M(N)))) \#* (\alpha\#W)$ **and**
 $(bn(p \cdot (\iota M(N)))) \#* (X @ A_Q @ xvec)$ **and** $(bn(p \cdot (\iota M(N)))) \#* (\Psi_Q\#Y)$

and $(bn(p \cdot (\iota M(N)))) \#* (Q\#(Q'\#(Q''\#Z)))$

by(*elim cBrComm1*(4)) (*assumption* | *simp*)+

then have *PeqP'*: $\Psi_P \otimes P\Psi' \simeq \Psi_{P'}$

and $A_{P'} \#* P'$ **and** $A_{P'} \#* (\iota M(N))$ **and** $A_{P'} \#* C$ **and** *distinct* $A_{P'}$

and $A_{P'} \#* Q$ **and** $A_{P'} \#* Q'$ **and** $A_{P'} \#* Q''$ **and** $A_{P'} \#* Z$ **and** $A_{P'} \#*$
 A_Q **and** $A_{P'} \#* X$ **and** $A_{P'} \#* \Psi_Q$ **and** $A_{P'} \#* Y$

and $A_{P'} \#* xvec$ **and** $A_{P'} \#* \alpha$ **and** $A_{P'} \#* (subject\ \alpha)$ **and** $A_{P'} \#* (bn\ \alpha)$
and $A_{P'} \#* (object\ \alpha)$

and $A_{P'} \#* V$ **and** $A_{P'} \#* W$

by(*simp del: freshChainSimps*)+

from *rPerm* $\langle bn\ \alpha \#* \Psi_P \rangle \langle xvec \#* \Psi_P \rangle$

have $(r \cdot \Psi_P) = \Psi_P$ **by** *simp*

from *PeqP'* **have** $r \cdot (\Psi_P \otimes P\Psi' \simeq \Psi_{P'})$

by *simp*

with $\langle r \cdot \Psi_P \rangle = \Psi_P$

have *rPeqP'*: $\Psi_P \otimes (r \cdot P\Psi') \simeq (r \cdot \Psi_{P'})$ **by**(*simp add: eqvts*)

from *rPerm* $\langle A_{P'} \#* (bn\ \alpha) \rangle \langle A_{P'} \#* xvec \rangle$

have $r \cdot A_{P'} = A_{P'}$ **by** *simp*

from *FrP'* **have** $r \cdot (extractFrame\ P' = \langle A_{P'}, \Psi_{P'} \rangle)$

by *simp*

with $\langle r \cdot A_{P'} = A_{P'} \rangle$ **have** *rFrP'*: $extractFrame\ (r \cdot P') = \langle A_{P'}, (r \cdot \Psi_{P'}) \rangle$

by(*simp add: eqvts*)

from $\langle xvec \#* \alpha \rangle$ **have** $(bn\ \alpha) \#* xvec$ **by** *simp*

from $\langle (A_P @ A_Q) \#* \alpha \rangle$ **have** $A_Q \#* bn\ \alpha$ **by** *simp*

from $\langle A_Q \#* M \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* (\iota M(\nu * xvec)(N))$ **by**

simp
from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec Q' \rangle \langle \text{xvec} \#* M \rangle \langle \text{distinct xvec} \rangle$
 $\langle \text{bn } \alpha \#* Q \rangle \langle \text{xvec} \#* \text{bn } \alpha \rangle$
have $\text{bn } \alpha \#* Q'$ **by** (*simp add: brotputFreshChainDerivative*)

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec Q' \rangle \langle \text{xvec} \#* M \rangle \langle \text{distinct xvec} \rangle$
 $\langle \text{bn } \alpha \#* Q \rangle \langle \text{bn } \alpha \#* \text{xvec} \rangle$
have $\text{bn } \alpha \#* N$ **by** (*simp add: brotputFreshChainDerivative*)

note $\langle \text{bn } \alpha \#* \text{subject } \alpha \rangle$

moreover from $\langle \text{bn } \alpha \#* (P \parallel Q) \rangle$ **have** $\text{bn } \alpha \#* Q$ **and** $\text{bn } \alpha \#* P$ **by** *simp+*
moreover from $\langle \text{bn } \alpha \#* V \rangle \langle \text{bn } \alpha \#* N \rangle$ **have** $\text{bn } \alpha \#* (N \# V)$ **by** *simp*
moreover note $\langle \text{bn } \alpha \#* \Psi_P \rangle \langle \text{bn } \alpha \#* \Psi_Q \rangle \langle \text{bn } \alpha \#* W \rangle$
moreover from $\langle \text{bn } \alpha \#* X \rangle \langle A_P \#* \text{bn } \alpha \rangle \langle A_{P'} \#* \text{bn } \alpha \rangle \langle \text{bn } \alpha \#* \text{xvec} \rangle$
have $\text{bn } \alpha \#* (X @ \text{xvec} @ A_P @ A_{P'})$ **by** *simp*
moreover from $\langle \text{bn } \alpha \#* Y \rangle \langle \text{bn } \alpha \#* \Psi_P \rangle$ **have** $\text{bn } \alpha \#* (\Psi_P \# Y)$ **by** *simp*
moreover from $\langle \text{bn } \alpha \#* Z \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* Q \rangle$ **have** $\text{bn } \alpha \#* (P \# Q \# Z)$
by *simp*
moreover from $\langle A_Q \#* V \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* (N \# V)$ **by** *simp*
moreover note $\langle A_Q \#* W \rangle$
moreover from $\langle A_Q \#* X \rangle \langle A_P \#* A_Q \rangle \langle A_{P'} \#* A_Q \rangle \langle A_Q \#* \text{xvec} \rangle$ **have** $A_Q \#* (X @ \text{xvec} @ A_P @ A_{P'})$ **by** *simp*
moreover from $\langle A_Q \#* Y \rangle \langle A_Q \#* \Psi_P \rangle$ **have** $A_Q \#* (\Psi_P \# Y)$ **by** *force*
moreover from $\langle A_Q \#* Z \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle$ **have** $A_Q \#* (P \# Q \# Z)$ **by** *simp*

moreover note $\langle \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec Q' = \alpha \prec Q'' \rangle$
ultimately obtain $q \ Q \Psi' \ A_Q' \ \Psi_Q'$ **where** $Sq: \text{set } q \subseteq \text{set } (\text{bn } \alpha) \times \text{set } (\text{bn } (q \cdot \alpha))$ **and** $QeqQ': ((q \cdot \Psi_Q) \otimes Q\Psi') \simeq \Psi_Q'$
and $\text{distinctPerm } q$ **and** $\text{bn}(q \cdot \alpha) \#* C'$ **and** $\text{Fr}Q': \text{extractFrame } Q'' = \langle A_Q', \Psi_Q' \rangle$
and $A_Q' \#* Q''$ **and** $A_Q' \#* \alpha$ **and** $A_Q' \#* (q \cdot \alpha)$ **and** $A_Q' \#* C$
and $\text{bn}(q \cdot \alpha) \#* \alpha$ **and** $\text{bn}(q \cdot \alpha) \#* Q''$ **and** $\text{distinct } A_Q'$
and $A_Q' \#* (N \# V)$ **and** $A_Q' \#* W$ **and** $A_Q' \#* (X @ \text{xvec} @ A_P @ A_{P'})$
and $A_Q' \#* (\Psi_P \# Y)$
and $A_Q' \#* (P \# Q \# Z)$ **and** $\text{bn}(q \cdot \alpha) \#* (N \# V)$ **and** $\text{bn}(q \cdot \alpha) \#* W$ **and**
 $\text{bn}(q \cdot \alpha) \#* (X @ \text{xvec} @ A_P @ A_{P'})$ **and** $\text{bn}(q \cdot \alpha) \#* (\Psi_P \# Y)$
and $\text{bn}(q \cdot \alpha) \#* (P \# Q \# Z)$
using $\langle A_Q \#* Q \rangle \langle A_Q \#* \alpha \rangle \langle A_Q \#* C \rangle \langle A_Q \#* C' \rangle$
 $\langle \text{bn } \alpha \#* C' \rangle \langle \alpha \neq \tau \rangle \langle A_Q \#* Q'' \rangle \langle \text{distinct } (\text{bn } \alpha) \rangle$
by (*elim cBrComm1 (8)*) [**where** $b=C$ **and** $ba=C'$ **and** $bb=\alpha$ **and** $bc=Q''$ **and**
 $bf=(X @ \text{xvec} @ A_P @ A_{P'})$ **and** $bg=(\Psi_P \# Y)$ **and** $bh=(P \# Q \# Z)$] (*assumption*
 $| \text{simp}$) +
then have $A_Q' \#* P$ **and** $A_Q' \#* Z$ **and** $A_Q' \#* A_P$ **and** $A_Q' \#* A_{P'}$ **and**
 $A_Q' \#* X$ **and** $A_Q' \#* \Psi_P$ **and** $A_Q' \#* Y$ **and** $A_Q' \#* Q$ **and** $A_Q' \#* N$
and $A_Q' \#* \text{xvec}$ **and** $A_Q' \#* V$ **and** $A_Q' \#* W$
and $\text{bn}(q \cdot \alpha) \#* A_P$ **and** $\text{bn}(q \cdot \alpha) \#* A_{P'}$ **and** $\text{bn}(q \cdot \alpha) \#* X$ **and** $\text{bn}(q \cdot \alpha) \#* Y$ **and** $\text{bn}(q \cdot \alpha) \#* Z$ **and** $\text{bn}(q \cdot \alpha) \#* \Psi_P$

and $bn(q \cdot \alpha) \#* P$ **and** $bn(q \cdot \alpha) \#* W$ **and** $bn(q \cdot \alpha) \#* V$ **and** $bn(q \cdot \alpha)$
 $\#* N$ **and** $bn(q \cdot \alpha) \#* xvec$
by(*simp del: freshChainSimps*)

from $\langle A_{P'} \#* Q'' \rangle \langle A_{Q'} \#* A_{P'} \rangle FrQ'$
have $A_{P'} \#* \Psi_{Q'}$
by(*metis extractFrameFreshChain freshFrameDest*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P' \rangle \langle A_{Q'} \#* P \rangle \langle A_{Q'} \#* N \rangle$
have $A_{Q'} \#* P'$
by(*simp add: brinputFreshChainDerivative*)
with $FrP' \langle A_{Q'} \#* A_{P'} \rangle$ **have** $A_{Q'} \#* \Psi_{P'}$
by(*metis extractFrameFreshChain freshFrameDest*)

have $(\Psi_P \otimes (r \cdot P\Psi')) \otimes ((q \cdot \Psi_Q) \otimes Q\Psi') \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by(*metis Composition' rPeqP' QeqQ'*)
then have $(\Psi_P \otimes ((r \cdot P\Psi') \otimes ((q \cdot \Psi_Q) \otimes Q\Psi'))) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity*)
then have $(\Psi_P \otimes (((q \cdot \Psi_Q) \otimes Q\Psi') \otimes (r \cdot P\Psi'))) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
then have $(\Psi_P \otimes ((q \cdot \Psi_Q) \otimes (Q\Psi' \otimes (r \cdot P\Psi')))) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity compositionSym*)
then have $((\Psi_P \otimes (q \cdot \Psi_Q)) \otimes (Q\Psi' \otimes (r \cdot P\Psi'))) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by (*metis AssertionStatEqTrans Associativity*)
then have $((\Psi_P \otimes (q \cdot \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
with $Sq \langle bn \alpha \#* \Psi_P \rangle \langle bn(q \cdot \alpha) \#* \Psi_P \rangle$ **have** $((q \cdot (\Psi_P \otimes \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'}$
by(*simp add: eqvts*)

from $Sq \langle A_{P'} \#* \alpha \rangle \langle bn(q \cdot \alpha) \#* A_{P'} \rangle$ **have** $A_{P'} \#* (q \cdot \alpha)$
by (*metis actionFreshChain freshChainSym freshStarChainSimps fresh-star-set-eq*)

from $\langle A_{Q'} \#* \alpha \rangle$ **have** $A_{Q'} \#* bn \alpha$ **by** *simp*
from $rPerm \langle A_{P'} \#* P' \rangle \langle A_{P'} \#* xvec \rangle \langle A_{P'} \#* bn \alpha \rangle$
have $A_{P'} \#* (r \cdot P')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from $rPerm \langle A_{Q'} \#* P' \rangle \langle A_{Q'} \#* xvec \rangle \langle A_{Q'} \#* bn \alpha \rangle$
have $A_{Q'} \#* (r \cdot P')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from $rPerm \langle A_{Q'} \#* \Psi_{P'} \rangle \langle A_{Q'} \#* xvec \rangle \langle A_{Q'} \#* bn \alpha \rangle$
have $A_{Q'} \#* (r \cdot \Psi_{P'})$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

with $\langle A_{P'} \#* \Psi_{Q'} \rangle \langle A_{Q'} \#* A_{P'} \rangle rFrP' FrQ'$
have $extractFrame ((r \cdot P') \parallel Q') = \langle (A_{P'} @ A_{Q'}), ((r \cdot \Psi_{P'}) \otimes \Psi_{Q'}) \rangle$

by simp
with $\langle PQ' = ((r \cdot P') \parallel Q'') \rangle$
have $\text{extractFrame } PQ' = \langle (A_{P'}' @ A_{Q'}'), ((r \cdot \Psi_{P'}) \otimes \Psi_{Q'}) \rangle$
by simp

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P' \rangle \langle \text{bn}(q \cdot \alpha) \#* P \rangle \langle \text{bn}(q \cdot \alpha) \#* N \rangle$
have $\text{bn}(q \cdot \alpha) \#* P'$ **by** (*simp add: brinputFreshChainDerivative*)

with $r\text{Perm } \langle \text{bn}(q \cdot \alpha) \#* \alpha \rangle \langle \text{bn}(q \cdot \alpha) \#* \text{vec} \rangle$
have $\text{bn}(q \cdot \alpha) \#* (r \cdot P')$
by (*metis actionFreshChain freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

from $\langle A_{P'}' \#* Q'' \rangle \langle A_{P'}' \#* (r \cdot P') \rangle \langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_{P'}' \#* PQ'$
by simp
from $\langle A_{Q'}' \#* Q'' \rangle \langle A_{Q'}' \#* (r \cdot P') \rangle \langle PQ' = (r \cdot P') \parallel Q'' \rangle$ **have** $A_{Q'}' \#* PQ'$
by simp

note $\langle \text{set } q \subseteq (\text{set } (\text{bn } \alpha)) \times \text{set } (\text{bn}(q \cdot \alpha)) \rangle$
 $\langle ((q \cdot (\Psi_P \otimes \Psi_Q)) \otimes ((r \cdot P\Psi') \otimes Q\Psi')) \simeq (r \cdot \Psi_{P'}) \otimes \Psi_{Q'} \rangle \langle \text{distinctPerm } q \rangle$
 $\langle \text{extractFrame } PQ' = \langle (A_{P'}' @ A_{Q'}'), ((r \cdot \Psi_{P'}) \otimes \Psi_{Q'}) \rangle \rangle$

moreover from $\langle A_{P'}' \#* PQ' \rangle \langle A_{Q'}' \#* PQ' \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* PQ'$ **by** *simp*

moreover from $\langle A_{P'}' \#* \alpha \rangle \langle A_{Q'}' \#* \alpha \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* \alpha$ **by** *simp*
moreover from $\langle A_{P'}' \#* (q \cdot \alpha) \rangle \langle A_{Q'}' \#* (q \cdot \alpha) \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* (q \cdot \alpha)$ **by** *simp*

moreover from $\langle A_{P'}' \#* C \rangle \langle A_{Q'}' \#* C \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* C$ **by** *simp*
moreover note $\langle \text{bn}(q \cdot \alpha) \#* C' \rangle \langle \text{bn}(q \cdot \alpha) \#* \alpha \rangle$
moreover from $\langle \text{bn}(q \cdot \alpha) \#* (r \cdot P') \rangle \langle \text{bn}(q \cdot \alpha) \#* Q'' \rangle \langle PQ' = (r \cdot P') \parallel Q'' \rangle$
have $\text{bn}(q \cdot \alpha) \#* PQ'$ **by** *simp*

moreover from $\langle A_{P'}' \#* V \rangle \langle A_{Q'}' \#* V \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* V$ **by** *simp*
moreover from $\langle A_{P'}' \#* W \rangle \langle A_{Q'}' \#* W \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* W$ **by** *simp*
moreover from $\langle A_{P'}' \#* X \rangle \langle A_{Q'}' \#* X \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* X$ **by** *simp*
moreover from $\langle A_{P'}' \#* Y \rangle \langle A_{Q'}' \#* Y \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* Y$ **by** *simp*
moreover from $\langle A_{P'}' \#* Z \rangle \langle A_{Q'}' \#* Z \rangle$ **have** $(A_{P'}' @ A_{Q'}') \#* Z$ **by** *simp*
moreover from $\langle \text{distinct } A_{P'}' \rangle \langle \text{distinct } A_{Q'}' \rangle \langle A_{Q'}' \#* A_{P'}' \rangle$
have $\text{distinct}(A_{P'}' @ A_{Q'}')$ **by** *simp*

moreover note $\langle \text{bn}(q \cdot \alpha) \#* V \rangle \langle \text{bn}(q \cdot \alpha) \#* W \rangle$
 $\langle \text{bn}(q \cdot \alpha) \#* X \rangle \langle \text{bn}(q \cdot \alpha) \#* Y \rangle \langle \text{bn}(q \cdot \alpha) \#* Z \rangle$
ultimately show *?case*
by (*rule cBrComm1(63)*)

next
case (*cBrComm2* $\Psi \Psi_Q P M \text{vec } N P' A_P \Psi_P Q Q' A_Q C C' \alpha PQ' V W X Y Z$)
have $\text{FrP: extractFrame } P = \langle A_P, \Psi_P \rangle$ **and** $\text{FrQ: extractFrame } Q = \langle A_Q, \Psi_Q \rangle$

by fact+
from $\langle xvec \#* \alpha \rangle$ **have** $xvec \#* bn \alpha$ **by simp**
from $\langle iM(\nu*xvec)\langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ **have** $\alpha \prec PQ' = iM(\nu*xvec)\langle N \rangle \prec P' \parallel Q'$ **by simp**
with $\langle xvec \#* (bn \alpha) \rangle$
obtain $P'' r$ **where** $rPerm: set r \subseteq set (bn \alpha) \times set xvec$
and $PQ' = P'' \parallel (r \cdot Q')$ **and** $\alpha \prec P'' = iM(\nu*xvec)\langle N \rangle \prec P'$
by $(elim\ actionPar1Dest) (assumption \mid simp)+$
then have $iM(\nu*xvec)\langle N \rangle \prec P' = \alpha \prec P''$ **by simp**

from $\langle A_Q \#* PQ' \rangle \langle PQ' = P'' \parallel (r \cdot Q') \rangle$ **have** $A_Q \#* P''$
by simp

from $\langle A_P \#* PQ' \rangle \langle PQ' = P'' \parallel (r \cdot Q') \rangle$ **have** $A_P \#* P''$
by simp

from $\langle bn \alpha \#* (P \parallel Q) \rangle \langle A_P \#* \alpha \rangle$
have $bn \alpha \#* P$ **and** $A_P \#* bn \alpha$ **by simp+**
with $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$ **have** $bn \alpha \#* \Psi_P$
by $(metis\ extractFrameFreshChain\ freshFrameDest)$

from $\langle bn \alpha \#* (P \parallel Q) \rangle \langle A_Q \#* \alpha \rangle$
have $bn \alpha \#* Q$ **and** $A_Q \#* bn \alpha$ **by simp+**
with $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $bn \alpha \#* \Psi_Q$
by $(metis\ extractFrameFreshChain\ freshFrameDest)$

from $rPerm \langle A_Q \#* xvec \rangle \langle xvec \#* \Psi_Q \rangle \langle A_Q \#* bn \alpha \rangle \langle bn \alpha \#* \Psi_Q \rangle$
have $r \cdot A_Q = A_Q$ **and** $r \cdot \Psi_Q = \Psi_Q$ **by simp+**

have $iM(\langle N \rangle) \prec Q' = iM(\langle N \rangle) \prec Q'$ **by simp**
moreover note $\langle A_Q \#* P \rangle \langle A_Q \#* C \rangle \langle A_Q \#* C' \rangle$
moreover from $\langle A_Q \#* M \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* (iM(\langle N \rangle))$ **by simp**
moreover note $\langle A_Q \#* V \rangle$
moreover from $\langle A_Q \#* W \rangle \langle A_Q \#* \alpha \rangle$ **have** $A_Q \#* (\alpha \# W)$ **by simp**
moreover from $\langle A_Q \#* X \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* xvec \rangle$ **have** $A_Q \#* (X @ A_P @ xvec)$
by simp
moreover from $\langle A_Q \#* Y \rangle \langle A_Q \#* \Psi_P \rangle$ **have** $A_Q \#* (\Psi_P \# Y)$ **by force**
moreover from $\langle A_Q \#* Z \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* P'' \rangle$ **have** $A_Q \#* (P \# (P' \# (P'' \# Z)))$ **by simp**
moreover note $\langle A_Q \#* Q' \rangle \langle A_Q \#* Q \rangle$

ultimately obtain $q Q \Psi' A_Q' \Psi_Q'$ **where** $Sq: set q \subseteq set (bn (iM(\langle N \rangle))) \times set (bn(q \cdot (iM(\langle N \rangle))))$ **and** $((q \cdot \Psi_Q) \otimes Q \Psi') \simeq \Psi_Q'$
and $distinctPerm\ q$ **and** $FrQ': extractFrame Q' = \langle A_Q', \Psi_Q' \rangle$
and $A_Q' \#* Q'$ **and** $A_Q' \#* (iM(\langle N \rangle))$ $A_Q' \#* (q \cdot (iM(\langle N \rangle)))$ **and** $A_Q' \#* C$
and $(bn(q \cdot (iM(\langle N \rangle)))) \#* C'$
and $(bn(q \cdot (iM(\langle N \rangle)))) \#* (iM(\langle N \rangle))$ **and** $(bn(q \cdot (iM(\langle N \rangle)))) \#* Q'$ **and**

distinct $A_{Q'}$
and $A_{Q'} \#* V$ **and** $A_{Q'} \#* (\alpha \# W)$ **and** $A_{Q'} \#* (X @ A_P @ xvec)$ **and** $A_{Q'} \#* (\Psi_P \# Y)$
and $A_{Q'} \#* (P \# (P' \# (P'' \# Z)))$ **and** $(bn(q \cdot (iM(N)))) \#* (\alpha \# W)$ **and**
 $(bn(q \cdot (iM(N)))) \#* (X @ A_P @ xvec)$ **and** $(bn(q \cdot (iM(N)))) \#* (\Psi_P \# Y)$
and $(bn(q \cdot (iM(N)))) \#* (P \# (P' \# (P'' \# Z)))$
by(*elim cBrComm2*(δ)) (*assumption* | *simp*)+

then have $QeqQ'$: $\Psi_Q \otimes Q\Psi' \simeq \Psi_{Q'}$
and $A_{Q'} \#* Q'$ **and** $A_{Q'} \#* (iM(N))$ **and** $A_{Q'} \#* C$ **and** *distinct* $A_{Q'}$
and $A_{Q'} \#* P$ **and** $A_{Q'} \#* P'$ **and** $A_{Q'} \#* P''$ **and** $A_{Q'} \#* Z$ **and** $A_{Q'} \#*$
 A_P **and** $A_{Q'} \#* X$ **and** $A_{Q'} \#* \Psi_P$ **and** $A_{Q'} \#* Y$
and $A_{Q'} \#* xvec$ **and** $A_{Q'} \#* \alpha$ **and** $A_{Q'} \#* (subject \alpha)$ **and** $A_{Q'} \#* (bn \alpha)$
and $A_{Q'} \#* (object \alpha)$
and $A_{Q'} \#* V$ **and** $A_{Q'} \#* W$
by(*simp del: freshChainSimps*)+

from $rPerm \langle bn \alpha \#* \Psi_Q \rangle \langle xvec \#* \Psi_Q \rangle$
have $(r \cdot \Psi_Q) = \Psi_Q$ **by** *simp*

from $QeqQ'$ **have** $r \cdot (\Psi_Q \otimes Q\Psi' \simeq \Psi_{Q'})$
by *simp*
with $\langle r \cdot \Psi_Q \rangle = \Psi_Q$
have $rQeqQ'$: $\Psi_Q \otimes (r \cdot Q\Psi') \simeq (r \cdot \Psi_Q)$ **by**(*simp add: eqvts*)

from $rPerm \langle A_{Q'} \#* (bn \alpha) \rangle \langle A_{Q'} \#* xvec \rangle$
have $r \cdot A_{Q'} = A_{Q'}$ **by** *simp*

from FrQ' **have** $r \cdot (extractFrame Q' = \langle A_{Q'}, \Psi_Q \rangle)$
by *simp*

with $\langle r \cdot A_{Q'} = A_{Q'} \rangle$ **have** $rFrQ'$: $extractFrame (r \cdot Q') = \langle A_{Q'}, (r \cdot \Psi_Q) \rangle$
by(*simp add: eqvts*)

from $\langle xvec \#* \alpha \rangle$ **have** $(bn \alpha) \#* xvec$ **by** *simp*

from $\langle (A_P @ A_Q) \#* \alpha \rangle$ **have** $A_P \#* bn \alpha$ **by** *simp*

from $\langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* N \rangle$ **have** $A_P \#* (iM(\nu * xvec) \langle N \rangle)$ **by**
simp
from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu * xvec) \langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle distinct \ xvec \rangle$
 $\langle bn \alpha \#* P \rangle \langle xvec \#* bn \alpha \rangle$
have $bn \alpha \#* P'$ **by**(*simp add: broutputFreshChainDerivative*)

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu * xvec) \langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle distinct \ xvec \rangle$
 $\langle bn \alpha \#* P \rangle \langle bn \alpha \#* xvec \rangle$
have $bn \alpha \#* N$ **by**(*simp add: broutputFreshChainDerivative*)

note $\langle bn \alpha \#* subject \alpha \rangle$

moreover from $\langle bn \ \alpha \ \#\# \ (P \parallel Q) \rangle$ **have** $bn \ \alpha \ \#\# \ Q$ **and** $bn \ \alpha \ \#\# \ P$ **by** *simp+*
moreover from $\langle bn \ \alpha \ \#\# \ V \rangle \langle bn \ \alpha \ \#\# \ N \rangle$ **have** $bn \ \alpha \ \#\# \ (N\#V)$ **by** *simp*
moreover note $\langle bn \ \alpha \ \#\# \ \Psi_P \rangle \langle bn \ \alpha \ \#\# \ \Psi_Q \rangle \langle bn \ \alpha \ \#\# \ W \rangle$
moreover from $\langle bn \ \alpha \ \#\# \ X \rangle \langle A_Q \ \#\# \ bn \ \alpha \rangle \langle A_{Q'} \ \#\# \ bn \ \alpha \rangle \langle bn \ \alpha \ \#\# \ xvec \rangle$
have $bn \ \alpha \ \#\# \ (X @ xvec @ A_Q @ A_{Q'})$ **by** *simp*
moreover from $\langle bn \ \alpha \ \#\# \ Y \rangle \langle bn \ \alpha \ \#\# \ \Psi_Q \rangle$ **have** $bn \ \alpha \ \#\# \ (\Psi_Q \# Y)$ **by** *simp*
moreover from $\langle bn \ \alpha \ \#\# \ Z \rangle \langle bn \ \alpha \ \#\# \ P \rangle \langle bn \ \alpha \ \#\# \ Q \rangle$ **have** $bn \ \alpha \ \#\# \ (P\#Q\#Z)$
by *simp*
moreover from $\langle A_P \ \#\# \ V \rangle \langle A_P \ \#\# \ N \rangle$ **have** $A_P \ \#\# \ (N\#V)$ **by** *simp*
moreover note $\langle A_P \ \#\# \ W \rangle$
moreover from $\langle A_P \ \#\# \ X \rangle \langle A_P \ \#\# \ A_Q \rangle \langle A_{Q'} \ \#\# \ A_P \rangle \langle A_P \ \#\# \ xvec \rangle$ **have** $A_P \ \#\# \ (X @ xvec @ A_Q @ A_{Q'})$ **by** *simp*
moreover from $\langle A_P \ \#\# \ Y \rangle \langle A_P \ \#\# \ \Psi_Q \rangle$ **have** $A_P \ \#\# \ (\Psi_Q \# Y)$ **by** *force*
moreover from $\langle A_P \ \#\# \ Z \rangle \langle A_P \ \#\# \ Q \rangle \langle A_P \ \#\# \ P \rangle$ **have** $A_P \ \#\# \ (P\#Q\#Z)$ **by** *simp*

moreover note $\langle iM(\nu * xvec) \langle N \rangle \prec P' = \alpha \prec P'' \rangle$
ultimately obtain $p \ P \Psi' \ A_{P'} \ \Psi_{P'}$ **where** Sp : *set* $p \subseteq \text{set} \ (bn \ \alpha) \times \text{set} \ (bn \ (p \cdot \alpha))$ **and** $PeqP'$: $((p \cdot \Psi_P) \otimes P \Psi') \simeq \Psi_{P'}$
and *distinctPerm* p **and** $bn(p \cdot \alpha) \ \#\# \ C'$ **and** FrP' : *extractFrame* $P'' = \langle A_{P'}, \Psi_{P'} \rangle$
and $A_{P'} \ \#\# \ P''$ **and** $A_{P'} \ \#\# \ \alpha$ **and** $A_{P'} \ \#\# \ (p \cdot \alpha)$ **and** $A_{P'} \ \#\# \ C$
and $bn(p \cdot \alpha) \ \#\# \ \alpha$ **and** $bn(p \cdot \alpha) \ \#\# \ P''$ **and** *distinct* $A_{P'}$
and $A_{P'} \ \#\# \ (N\#V)$ **and** $A_{P'} \ \#\# \ W$ **and** $A_{P'} \ \#\# \ (X @ xvec @ A_Q @ A_{Q'})$
and $A_{P'} \ \#\# \ (\Psi_Q \# Y)$
and $A_{P'} \ \#\# \ (P\#Q\#Z)$ **and** $bn(p \cdot \alpha) \ \#\# \ (N\#V)$ **and** $bn(p \cdot \alpha) \ \#\# \ W$ **and**
 $bn(p \cdot \alpha) \ \#\# \ (X @ xvec @ A_Q @ A_{Q'})$ **and** $bn(p \cdot \alpha) \ \#\# \ (\Psi_Q \# Y)$
and $bn(p \cdot \alpha) \ \#\# \ (P\#Q\#Z)$
using $\langle A_P \ \#\# \ P \rangle \langle A_P \ \#\# \ \alpha \rangle \langle A_P \ \#\# \ C \rangle \langle A_P \ \#\# \ C' \rangle$
 $\langle bn \ \alpha \ \#\# \ C' \rangle \langle \alpha \neq \tau \rangle \langle A_P \ \#\# \ P'' \rangle \langle \text{distinct} \ (bn \ \alpha) \rangle$
by (*elim cBrComm2*(4)[**where** $b=C$ **and** $ba=C'$ **and** $bb=\alpha$ **and** $bc=P''$ **and**
 $bf=(X @ xvec @ A_Q @ A_{Q'})$ **and** $bg=(\Psi_Q \# Y)$ **and** $bh=(P\#Q\#Z)$]) (*assumption*
| *simp*)
then **have** $A_{P'} \ \#\# \ Q$ **and** $A_{P'} \ \#\# \ Z$ **and** $A_{P'} \ \#\# \ A_Q$ **and** $A_{Q'} \ \#\# \ A_{P'}$ **and**
 $A_{P'} \ \#\# \ X$ **and** $A_{P'} \ \#\# \ \Psi_Q$ **and** $A_{P'} \ \#\# \ Y$ **and** $A_{P'} \ \#\# \ P$ **and** $A_{P'} \ \#\# \ N$
and $A_{P'} \ \#\# \ xvec$ **and** $A_{P'} \ \#\# \ V$ **and** $A_{P'} \ \#\# \ W$
and $bn(p \cdot \alpha) \ \#\# \ A_Q$ **and** $bn(p \cdot \alpha) \ \#\# \ A_{Q'}$ **and** $bn(p \cdot \alpha) \ \#\# \ X$ **and** $bn(p \cdot \alpha) \ \#\# \ Y$
and $bn(p \cdot \alpha) \ \#\# \ Z$ **and** $bn(p \cdot \alpha) \ \#\# \ \Psi_Q$
and $bn(p \cdot \alpha) \ \#\# \ Q$ **and** $bn(p \cdot \alpha) \ \#\# \ W$ **and** $bn(p \cdot \alpha) \ \#\# \ V$ **and** $bn(p \cdot \alpha) \ \#\# \ N$
and $bn(p \cdot \alpha) \ \#\# \ xvec$
by (*simp del: freshChainSimps*)

from $\langle A_{Q'} \ \#\# \ P'' \rangle \langle A_{Q'} \ \#\# \ A_{P'} \rangle$ FrP'
have $A_{Q'} \ \#\# \ \Psi_{P'}$
by (*metis extractFrameFreshChain freshFrameDest*)

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q' \rangle \langle A_{P'} \ \#\# \ Q \rangle \langle A_{P'} \ \#\# \ N \rangle$
have $A_{P'} \ \#\# \ Q'$

by (*simp add: brinputFreshChainDerivative*)
with $\langle FrQ' \langle A_Q' \#* A_P' \rangle \text{ have } A_P' \#* \Psi_Q' \rangle$
by (*metis extractFrameFreshChain freshFrameDest*)

have $((p \cdot \Psi_P) \otimes P\Psi') \otimes (\Psi_Q \otimes (r \cdot Q\Psi')) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis Composition' PeqP' rQeqQ'*)
then have $((p \cdot \Psi_P) \otimes (P\Psi' \otimes (\Psi_Q \otimes (r \cdot Q\Psi')))) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity*)
then have $((p \cdot \Psi_P) \otimes ((\Psi_Q \otimes (r \cdot Q\Psi')) \otimes P\Psi')) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
then have $(p \cdot \Psi_P) \otimes (\Psi_Q \otimes ((r \cdot Q\Psi') \otimes P\Psi')) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity compositionSym*)
then have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes ((r \cdot Q\Psi') \otimes P\Psi') \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis AssertionStatEqTrans Associativity*)
then have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (P\Psi' \otimes (r \cdot Q\Psi')) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'})$
by (*metis AssertionStatEqSym AssertionStatEqTrans Associativity associativitySym*)
with $\langle Sp \langle bn \alpha \#* \Psi_Q \rangle \langle bn (p \cdot \alpha) \#* \Psi_Q \rangle \text{ have } (p \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes (r \cdot Q\Psi')) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'}) \rangle$
by (*simp add: eqvts*)

from $\langle Sp \langle A_Q' \#* \alpha \rangle \langle bn(p \cdot \alpha) \#* A_Q' \rangle \text{ have } A_Q' \#* (p \cdot \alpha) \rangle$
by (*metis actionFreshChain freshChainSym freshStarChainSimps fresh-star-set-eq*)

from $\langle A_P' \#* \alpha \rangle$ **have** $A_P' \#* bn \alpha$ **by** *simp*
from $rPerm \langle A_Q' \#* Q' \rangle \langle A_Q' \#* xvec \rangle \langle A_Q' \#* bn \alpha \rangle$
have $A_Q' \#* (r \cdot Q')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from $rPerm \langle A_P' \#* Q' \rangle \langle A_P' \#* xvec \rangle \langle A_P' \#* bn \alpha \rangle$
have $A_P' \#* (r \cdot Q')$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)
from $rPerm \langle A_P' \#* \Psi_{Q'} \rangle \langle A_P' \#* xvec \rangle \langle A_P' \#* bn \alpha \rangle$
have $A_P' \#* (r \cdot \Psi_{Q'})$
by (*metis freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

with $\langle A_Q' \#* \Psi_{P'} \rangle \langle A_Q' \#* A_P' \rangle FrP' rFrQ'$
have $extractFrame (P'' \parallel (r \cdot Q')) = \langle (A_P' @ A_Q'), (\Psi_{P'} \otimes (r \cdot \Psi_{Q'})) \rangle$
by *simp*
with $\langle PQ' = (P'' \parallel (r \cdot Q')) \rangle$
have $extractFrame PQ' = \langle (A_P' @ A_Q'), (\Psi_{P'} \otimes (r \cdot \Psi_{Q'})) \rangle$
by *simp*

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto ;M(|N|) \prec Q' \rangle \langle bn(p \cdot \alpha) \#* Q \rangle \langle bn(p \cdot \alpha) \#* N \rangle$
have $bn(p \cdot \alpha) \#* Q'$ **by** (*simp add: brinputFreshChainDerivative*)

with $rPerm \langle bn(p \cdot \alpha) \#* \alpha \rangle \langle bn(p \cdot \alpha) \#* xvec \rangle$
have $bn(p \cdot \alpha) \#* (r \cdot Q')$

by (*metis actionFreshChain freshAlphaPerm freshChainSym name-list-set-fresh permStarFresh*)

from $\langle A_{P'} \# P'' \rangle \langle A_{P'} \# (r \cdot Q') \rangle \langle PQ' = P'' \parallel (r \cdot Q') \rangle$ **have** $A_{P'} \# PQ'$
by *simp*

from $\langle A_{Q'} \# P'' \rangle \langle A_{Q'} \# (r \cdot Q') \rangle \langle PQ' = P'' \parallel (r \cdot Q') \rangle$ **have** $A_{Q'} \# PQ'$
by *simp*

note $\langle \text{set } p \subseteq (\text{set } (bn \ \alpha)) \times \text{set } (bn(p \cdot \alpha)) \rangle$
 $\langle ((p \cdot (\Psi_P \otimes \Psi_Q)) \otimes (P\Psi' \otimes (r \cdot Q\Psi'))) \simeq \Psi_{P'} \otimes (r \cdot \Psi_{Q'}) \rangle \langle \text{distinctPerm } p \rangle$
 $\langle \text{extractFrame } PQ' = \langle (A_{P'} @ A_{Q'}), (\Psi_{P'} \otimes (r \cdot \Psi_{Q'})) \rangle \rangle$

moreover from $\langle A_{P'} \# PQ' \rangle \langle A_{Q'} \# PQ' \rangle$ **have** $(A_{P'} @ A_{Q'}) \# PQ'$ **by** *simp*

moreover from $\langle A_{P'} \# \alpha \rangle \langle A_{Q'} \# \alpha \rangle$ **have** $(A_{P'} @ A_{Q'}) \# \alpha$ **by** *simp*
moreover from $\langle A_{P'} \# (p \cdot \alpha) \rangle \langle A_{Q'} \# (p \cdot \alpha) \rangle$ **have** $(A_{P'} @ A_{Q'}) \# (p \cdot \alpha)$ **by** *simp*

moreover from $\langle A_{P'} \# C \rangle \langle A_{Q'} \# C \rangle$ **have** $(A_{P'} @ A_{Q'}) \# C$ **by** *simp*
moreover note $\langle bn(p \cdot \alpha) \# C' \rangle \langle bn(p \cdot \alpha) \# \alpha \rangle$
moreover from $\langle bn(p \cdot \alpha) \# (r \cdot Q') \rangle \langle bn(p \cdot \alpha) \# P'' \rangle \langle PQ' = P'' \parallel (r \cdot Q') \rangle$
have $bn(p \cdot \alpha) \# PQ'$ **by** *simp*

moreover from $\langle A_{P'} \# V \rangle \langle A_{Q'} \# V \rangle$ **have** $(A_{P'} @ A_{Q'}) \# V$ **by** *simp*
moreover from $\langle A_{P'} \# W \rangle \langle A_{Q'} \# W \rangle$ **have** $(A_{P'} @ A_{Q'}) \# W$ **by** *simp*
moreover from $\langle A_{P'} \# X \rangle \langle A_{Q'} \# X \rangle$ **have** $(A_{P'} @ A_{Q'}) \# X$ **by** *simp*
moreover from $\langle A_{P'} \# Y \rangle \langle A_{Q'} \# Y \rangle$ **have** $(A_{P'} @ A_{Q'}) \# Y$ **by** *simp*
moreover from $\langle A_{P'} \# Z \rangle \langle A_{Q'} \# Z \rangle$ **have** $(A_{P'} @ A_{Q'}) \# Z$ **by** *simp*
moreover from $\langle \text{distinct } A_{P'} \rangle \langle \text{distinct } A_{Q'} \rangle \langle A_{Q'} \# A_{P'} \rangle$
have $\text{distinct}(A_{P'} @ A_{Q'})$ **by** *simp*
moreover note $\langle bn(p \cdot \alpha) \# V \rangle \langle bn(p \cdot \alpha) \# W \rangle$
 $\langle bn(p \cdot \alpha) \# X \rangle \langle bn(p \cdot \alpha) \# Y \rangle \langle bn(p \cdot \alpha) \# Z \rangle$
ultimately show *?case*
by(*rule cBrComm2(63)*)

next
case *cBrClose*
then show *?case*
by(*simp add: residualInject*)

next
case(*cOpen* Ψ P M $xvec1$ $xvec2$ N P' x A_P Ψ_P C C' α P'' V W X Y Z)
from $\langle M(\nu*(xvec1 @ x\#xvec2)) \rangle \langle N \rangle \prec P' = \alpha \prec P'' \rangle \langle x \# xvec1 \rangle \langle x \# xvec2 \rangle$
 $\langle x \# \alpha \rangle \langle x \# P'' \rangle \langle \text{distinct}(bn \ \alpha) \rangle \langle A_P \# \alpha \rangle \langle x \# \alpha \rangle$
obtain $yvec1$ y $yvec2$ N' **where** $yvecEq: bn \ \alpha = yvec1 @ y\#yvec2$ **and**
 $P'eqP'': (\nu*(xvec1 @ xvec2)) \langle N \rangle \prec' P' = (\nu*(yvec1 @ yvec2)) \langle [(x, y)] \cdot N' \rangle \prec' \langle [(x, y)] \cdot P'' \rangle$ **and** $A_P \# N'$ **and** $Subj: \text{subject } \alpha = \text{Some } M$ **and** $x \# N'$ **and** $\alpha eq: \alpha = M(\nu*(yvec1 @ y\#yvec2)) \langle N' \rangle$
apply(*cases rule: actionCases[where $\alpha = \alpha$]*)
apply(*simp add: residualInject*)
apply(*simp add: residualInject*)

apply(simp add: residualInject)
apply(metis boundOutputOpenDest)
apply(simp add: residualInject)
by(simp add: residualInject)

note $\langle A_P \#* P \rangle \langle A_P \#* M \rangle$
moreover from *Subj yvecEq* $\langle bn \ \alpha \ \#* \ \text{subject } \alpha \rangle$ **have** $yvec1 \ \#* \ M \ yvec2 \ \#*$
M **by** *simp+*
moreover from *yvecEq* $\langle A_P \ \#* \ \alpha \rangle$ **have** $A_P \ \#* \ (yvec1 @ yvec2)$ **by** *simp*
moreover note $\langle A_P \ \#* \ C \rangle$
moreover from *yvecEq* $\langle bn \ \alpha \ \#* \ (\nu x)P \rangle \langle x \ \# \ \alpha \rangle$ **have** $(yvec1 @ yvec2) \ \#* \ P$
by *simp*
moreover from *yvecEq* $\langle bn \ \alpha \ \#* \ C' \rangle \langle bn \ \alpha \ \#* \ V \rangle \langle bn \ \alpha \ \#* \ W \rangle \langle bn \ \alpha \ \#* \ X \rangle$
 $\langle bn \ \alpha \ \#* \ Y \rangle \langle bn \ \alpha \ \#* \ Z \rangle \langle distinct(bn \ \alpha) \rangle \langle x \ \# \ \alpha \rangle$
have $(yvec1 @ yvec2) \ \#* \ C'$ **and** $(yvec1 @ yvec2) \ \#* \ V$ **and** $(yvec1 @ yvec2) \ \#* \ W$
and $(yvec1 @ yvec2) \ \#* \ (x \# y \# X)$ **and** $(yvec1 @ yvec2) \ \#* \ Y$ **and** $(yvec1 @ yvec2) \ \#* \ Z$
by *simp+*
moreover note $\langle A_P \ \#* \ V \rangle \langle A_P \ \#* \ W \rangle$
moreover from $\langle A_P \ \#* \ X \rangle \langle x \ \# \ A_P \rangle \langle A_P \ \#* \ \alpha \rangle$ *yvecEq* **have** $A_P \ \#* \ (x \# y \# X)$
by *simp*
moreover note $\langle A_P \ \#* \ Y \rangle \langle A_P \ \#* \ Z \rangle$
moreover from $\langle A_P \ \#* \ N' \rangle \langle A_P \ \#* \ P'' \rangle \langle x \ \# \ A_P \rangle \langle A_P \ \#* \ \alpha \rangle$ *yvecEq* **have**
 $A_P \ \#* \ ([(x, y)] \cdot N')$ **and** $A_P \ \#* \ ([(x, y)] \cdot P'')$
by *simp+*
moreover from *yvecEq* $\langle distinct(bn \ \alpha) \rangle$ **have** $distinct(yvec1 @ yvec2)$ **by** *simp*
moreover from $P' eq P''$ **have** $M(\nu*(xvec1 @ xvec2)) \langle N \rangle \prec P' = M(\nu*(yvec1 @ yvec2)) \langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot P''$
by(simp add: residualInject)
ultimately obtain $p \ \Psi' \ A_{P'} \ \Psi_{P'}$ **where** S : *set* $p \subseteq \text{set } (yvec1 @ yvec2) \times \text{set } (p \cdot (yvec1 @ yvec2))$ **and** $P_{eq} P'$: $((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_{P'}$
and $distinctPerm \ p$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ C'$ **and** FrP' : *extract-Frame* $([(x, y)] \cdot P'') = \langle A_{P'}, \Psi_{P'} \rangle$
and $A_{P'} \ \#* \ ([(x, y)] \cdot P'')$ **and** $A_{P'} \ \#* \ ([(x, y)] \cdot N')$ **and** $A_{P'} \ \#* \ C$ **and**
 $(p \cdot (yvec1 @ yvec2)) \ \#* \ ([(x, y)] \cdot N')$ **and** $A_{P'} \ \#* \ M$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#*$
 $(yvec1 @ yvec2)$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ M$ **and** $distinct \ A_{P'}$
and $(p \cdot (yvec1 @ yvec2)) \ \#* \ ([(x, y)] \cdot P'')$ **and** $(yvec1 @ yvec2) \ \#* \ A_{P'}$ **and**
 $(p \cdot (yvec1 @ yvec2)) \ \#* \ A_{P'}$
and $A_{P'} \ \#* \ V$ **and** $A_{P'} \ \#* \ W$ **and** $A_{P'} \ \#* \ (x \# y \# X)$ **and** $A_{P'} \ \#* \ Y$ **and**
 $A_{P'} \ \#* \ Z$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ (x \# y \# X)$
and $(p \cdot (yvec1 @ yvec2)) \ \#* \ V$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ W$ **and** $(p \cdot$
 $(yvec1 @ yvec2)) \ \#* \ Y$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ Z$ **using** $\langle A_P \ \#* \ C' \rangle$
by(*elim cOpen(4)*[**where** $b=C$ **and** $ba=C'$ **and** $bd=V$ **and** $be=W$ **and**
 $bf=x \# y \# X$ **and** $bg=Y$ **and** $bh=Z$]) (*assumption* | *simp*)+

from $\langle A_{P'} \ \#* \ (x \# y \# X) \rangle$ **have** $x \ \# \ A_{P'}$ **and** $y \ \# \ A_{P'}$ **and** $A_{P'} \ \#* \ X$ **by** *simp+*
from $\langle (p \cdot (yvec1 @ yvec2)) \ \#* \ (x \# y \# X) \rangle$ **have** $x \ \# \ (p \cdot (yvec1 @ yvec2))$ **and**
 $y \ \# \ (p \cdot (yvec1 @ yvec2))$ **and** $(p \cdot (yvec1 @ yvec2)) \ \#* \ X$ **by** *simp+*

from $\langle x \# \alpha \rangle$ *yvecEq* **have** $x \# yvec1$ **and** $x \neq y$ **and** $x \# yvec2$ **by** *simp+*
from $\langle distinct(bn \ \alpha) \rangle$ *yvecEq* **have** $yvec1 \#* yvec2$ **and** $y \# yvec1$ **and** $y \# yvec2$ **by** *simp+*
from $\langle bn \ \alpha \#* C' \rangle$ *yvecEq* **have** $yvec1 \#* C'$ **and** $y \# C'$ **and** $yvec2 \#* C'$ **by** *simp+*

from $S \langle x \# \alpha \rangle \langle x \# p \cdot (yvec1 @ yvec2) \rangle$ *yvecEq* **have** $x \# p$ **by** (*intro freshAlphaSwap*) (*assumption* | *simp*)
from $S \langle distinct(bn \ \alpha) \rangle \langle y \# p \cdot (yvec1 @ yvec2) \rangle$ *yvecEq* **have** $y \# p$ **by** (*intro freshAlphaSwap*) (*assumption* | *simp*)

from *yvecEq* $S \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \cdot (yvec1 @ yvec2) \rangle \langle y \# p \cdot (yvec1 @ yvec2) \rangle$
have $set((y, x) \# p) \subseteq set(bn \ \alpha) \times set(bn((y, x) \# p) \cdot \alpha)$
apply (*simp add: bnEqvt[symmetric]*)
by (*auto simp add: eqts calc-atm*)

moreover from *PeqP'* **have** $([(y, x)] \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq [(y, x)] \cdot \Psi_{P'}$
by (*simp add: AssertionStatEqClosed*)
then have $((y, x) \# p) \cdot \Psi_P \otimes [(y, x)] \cdot \Psi' \simeq [(y, x)] \cdot \Psi_{P'}$
by (*simp add: eqts*)

moreover from $\langle distinctPerm \ p \rangle$ $S \langle x \neq y \rangle \langle x \# p \rangle \langle y \# p \rangle$ **have** *distinctPerm* $((y, x) \# p)$
by *simp*

moreover from *FrP'* **have** $([(x, y)] \cdot (extractFrame([(x, y)] \cdot P'')) = ([(x, y)] \cdot \langle A_{P'}, \Psi_{P'} \rangle)$
by *simp*

with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $extractFrame \ P'' = \langle A_{P'}, [(y, x)] \cdot \Psi_{P'} \rangle$
by (*simp add: eqts name-swap*)

moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# N' \rangle \langle p \cdot (yvec1 @ yvec2) \rangle \#* ([(x, y)] \cdot N')$ **have** $([(y, x)] \cdot p \cdot (yvec1 @ yvec2)) \#* ([(y, x)] \cdot [(x, y)] \cdot N')$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

then have $((y, x) \# p) \cdot (yvec1 @ yvec2) \#* N'$ **by** (*simp add: name-swap*)

with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# C \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle x \# N' \rangle$ *yvecEq*
have $bn(((y, x) \# p) \cdot \alpha) \#* N'$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: eqts perm-compose calc-atm freshChainSimps*)

moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# N' \rangle \langle p \cdot (yvec1 @ yvec2) \rangle \#* ([(x, y)] \cdot P'')$ **have** $([(y, x)] \cdot p \cdot (yvec1 @ yvec2)) \#* ([(y, x)] \cdot [(x, y)] \cdot P'')$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

then have $((y, x) \# p) \cdot (yvec1 @ yvec2) \#* P''$ **by** (*simp add: name-swap*)

with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle x \# P'' \rangle$ *yvecEq*
have $bn(((y, x) \# p) \cdot \alpha) \#* P''$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqts freshChainSimps*)

moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# A_{P'} \rangle \langle p \cdot (yvec1 @ yvec2) \rangle \#* A_{P'}$ **have** $(p \cdot (yvec1 @ x \# yvec2)) \#* A_{P'}$
by (*simp add: eqts freshChainSimps*)

then have $([(y, x)] \cdot p \cdot (yvec1 @ x \# yvec2)) \#* ([(y, x)] \cdot A_{P'})$

by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# A_P' \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle y \# A_P' \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* A_P'$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# C' \rangle \langle (p \cdot (yvec1@yvec2)) \#* C' \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* C'$
by(*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* ([(y, x)] \cdot C')$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# C' \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle y \# C' \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* C'$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# X \rangle \langle (p \cdot (yvec1@yvec2)) \#* X \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* X$
by(*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* ([(y, x)] \cdot X)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# X \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle bn \alpha \#* X \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* X$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# V \rangle \langle (p \cdot (yvec1@yvec2)) \#* V \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* V$
by(*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* ([(y, x)] \cdot V)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# V \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle bn \alpha \#* V \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* V$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# W \rangle \langle (p \cdot (yvec1@yvec2)) \#* W \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* W$
by(*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* ([(y, x)] \cdot W)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# W \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle bn \alpha \#* W \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* W$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# Y \rangle \langle (p \cdot (yvec1@yvec2)) \#* Y \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* Y$
by(*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* ([(y, x)] \cdot Y)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# Y \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle \langle bn \alpha \#* Y \rangle yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \#* Y$ **by**(*simp add: bnEqvt*[*symmetric*]) (*simp add:*

perm-compose calc-atm eqvts freshChainSimps)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# Z \rangle \langle (p \cdot (yvec1 @ yvec2)) \#* Z \rangle$ **have** $(p \cdot (yvec1 @ x \# yvec2)) \#* Z$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1 @ x \# yvec2)) \#* ([(y, x)] \cdot Z)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# Z \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle bn \alpha \#* Z \rangle yvecEq$
have $bn(((y, x) \# p) \cdot \alpha) \#* Z$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle (yvec1 @ yvec2) \#* A_{P'} \rangle \langle y \# A_{P'} \rangle yvecEq$ **have** $bn \alpha \#* A_{P'}$
by *simp*
moreover from $\langle A_{P'} \#* ([(x, y)] \cdot N') \rangle$ **have** $([(x, y)] \cdot A_{P'}) \#* ([(x, y)] \cdot [(x, y)] \cdot N')$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $A_{P'} \#* N'$ **by** *simp*
with $\langle A_{P'} \#* M \rangle \langle (yvec1 @ yvec2) \#* A_{P'} \rangle \langle y \# A_{P'} \rangle \alpha eq$ **have** $A_{P'} \#* \alpha$ **by** *simp*
moreover then have $(((y, x) \# p) \cdot A_{P'}) \#* (((y, x) \# p) \cdot \alpha)$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle S \langle (yvec1 @ yvec2) \#* A_{P'} \rangle \langle (p \cdot (yvec1 @ yvec2)) \#* A_{P'} \rangle$
have $A_{P'} \#* (((y, x) \# p) \cdot \alpha)$ **by** (*simp add: eqvts*)
moreover from $\langle A_{P'} \#* ([(x, y)] \cdot P'') \rangle$ **have** $([(x, y)] \cdot A_{P'}) \#* ([(x, y)] \cdot [(x, y)] \cdot P'')$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $A_{P'} \#* P''$ **by** *simp*
moreover from $yvecEq \alpha eq \langle (p \cdot (yvec1 @ yvec2)) \#* (yvec1 @ yvec2) \rangle \langle y \# p \rangle$
 $\langle x \# \alpha \rangle S \langle (p \cdot (yvec1 @ yvec2)) \#* M \rangle \langle (p \cdot (yvec1 @ yvec2)) \#* ([(x, y)] \cdot N') \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
have $bn(((y, x) \# p) \cdot \alpha) \#* \alpha$
apply (*simp add: eqvts del: set-append*)
apply (*intro conjI*)
apply (*simp add: perm-compose eqvts del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply (*simp add: perm-compose eqvts del: set-append*)
apply (*simp add: perm-compose eqvts swapStarFresh del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) swapStarFresh calc-atm eqvts del: set-append*)
apply (*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply (*subst pt-fresh-star-bij[symmetric, OF pt-name-inst, OF at-name-inst,*

where $pi=[(x, y)]$
apply(*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply(*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
apply(*subst pt-fresh-star-bij[symmetric, OF pt-name-inst, OF at-name-inst,*
where $pi=[(x, y)]$)
by(*simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append*)
moreover note $\langle A_P' \#* C \rangle \langle A_P' \#* V \rangle \langle A_P' \#* W \rangle \langle A_P' \#* X \rangle \langle A_P' \#* Y \rangle$
 $\langle A_P' \#* Z \rangle \langle distinct A_P' \rangle$

ultimately show *?case*
by(*elim cOpen*)
next
case(*cBrOpen $\Psi P M xvec1 xvec2 N P' x A_P \Psi_P C C' \alpha P'' V W X Y Z$*)
from $\langle \downarrow M(\nu*(xvec1@x\#xvec2)) \rangle \langle N \rangle \prec P' = \alpha \prec P'' \langle x \# xvec1 \rangle \langle x \# xvec2 \rangle$
 $\langle x \# \alpha \rangle \langle x \# P'' \rangle \langle distinct(bn \alpha) \rangle \langle A_P \#* \alpha \rangle \langle x \# \alpha \rangle$
obtain $yvec1 y yvec2 N'$ **where** $yvecEq: bn \alpha = yvec1@y\#yvec2$ **and**
 $P'eqP'': (\nu*(xvec1@xvec2))N \prec' P' = (\nu*(yvec1@yvec2))([(x, y)] \cdot N') \prec' ([(x,$
 $y)] \cdot P'')$ **and** $A_P \#* N'$ **and** $Subj: subject \alpha = Some M$ **and** $x \# N'$ **and** $\alpha eq: \alpha$
 $= \downarrow M(\nu*(yvec1@y\#yvec2)) \langle N' \rangle$
apply(*cases rule: actionCases[where $\alpha=\alpha$]*)
apply(*simp-all add: residualInject*)
by (*metis boundOutputOpenDest*)

note $\langle A_P \#* P \rangle \langle A_P \#* M \rangle$
moreover from $Subj yvecEq \langle bn \alpha \#* subject \alpha \rangle$ **have** $yvec1 \#* M yvec2 \#*$
 M **by** *simp+*
moreover from $yvecEq \langle A_P \#* \alpha \rangle$ **have** $A_P \#* (yvec1@yvec2)$ **by** *simp*
moreover note $\langle A_P \#* C \rangle$
moreover from $yvecEq \langle bn \alpha \#* (\nu x)P \rangle \langle x \# \alpha \rangle$ **have** $(yvec1@yvec2) \#* P$
by *simp*
moreover from $yvecEq \langle bn \alpha \#* C' \rangle \langle bn \alpha \#* V \rangle \langle bn \alpha \#* W \rangle \langle bn \alpha \#* X \rangle$
 $\langle bn \alpha \#* Y \rangle \langle bn \alpha \#* Z \rangle \langle distinct(bn \alpha) \rangle \langle x \# \alpha \rangle$
have $(yvec1@yvec2) \#* C'$ **and** $(yvec1@yvec2) \#* V$ **and** $(yvec1@yvec2) \#* W$
and $(yvec1@yvec2) \#* (x\#y\#X)$ **and** $(yvec1@yvec2) \#* Y$ **and** $(yvec1@yvec2) \#*$
 Z
by *simp+*
moreover note $\langle A_P \#* V \rangle \langle A_P \#* W \rangle$
moreover from $\langle A_P \#* X \rangle \langle x \# A_P \rangle \langle A_P \#* \alpha \rangle yvecEq$ **have** $A_P \#* (x\#y\#X)$
by *simp*
moreover note $\langle A_P \#* Y \rangle \langle A_P \#* Z \rangle$
moreover from $\langle A_P \#* N' \rangle \langle A_P \#* P'' \rangle \langle x \# A_P \rangle \langle A_P \#* \alpha \rangle yvecEq$ **have**
 $A_P \#* ([(x, y)] \cdot N')$ **and** $A_P \#* ([(x, y)] \cdot P'')$
by *simp+*
moreover from $yvecEq \langle distinct(bn \alpha) \rangle$ **have** $distinct(yvec1@yvec2)$ **by** *simp*
moreover from $P'eqP''$ **have** $\downarrow M(\nu*(xvec1@xvec2)) \langle N \rangle \prec P' = \downarrow M(\nu*(yvec1@yvec2)) \langle ([(x,$
 $y)] \cdot N' \rangle \prec ([(x, y)] \cdot P'')$
by(*simp add: residualInject*)

ultimately obtain $p \Psi' A_{P'} \Psi_{P'}$ **where** $S: \text{set } p \subseteq \text{set } (yvec1@yvec2) \times \text{set } (p \cdot (yvec1@yvec2))$ **and** $PeqP': ((p \cdot \Psi_P) \otimes \Psi') \simeq \Psi_{P'}$
and $\text{distinctPerm } p$ **and** $(p \cdot (yvec1@yvec2)) \#* C'$ **and** FrP' : $\text{extractFrame}([(x, y)] \cdot P'') = \langle A_{P'}, \Psi_{P'} \rangle$
and $A_{P'} \#* ([x, y] \cdot P'')$ **and** $A_{P'} \#* ([x, y] \cdot N')$ **and** $A_{P'} \#* C$ **and** $(p \cdot (yvec1@yvec2)) \#* ([x, y] \cdot N')$ **and** $A_{P'} \#* M$ **and** $(p \cdot (yvec1@yvec2)) \#* (yvec1@yvec2)$ **and** $(p \cdot (yvec1@yvec2)) \#* M$ **and** $\text{distinct } A_{P'}$
and $(p \cdot (yvec1@yvec2)) \#* ([x, y] \cdot P'')$ **and** $(yvec1@yvec2) \#* A_{P'}$ **and** $(p \cdot (yvec1@yvec2)) \#* A_{P'}$
and $A_{P'} \#* V$ **and** $A_{P'} \#* W$ **and** $A_{P'} \#* (x\#y\#X)$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* Z$ **and** $(p \cdot (yvec1@yvec2)) \#* (x\#y\#X)$
and $(p \cdot (yvec1@yvec2)) \#* V$ **and** $(p \cdot (yvec1@yvec2)) \#* W$ **and** $(p \cdot (yvec1@yvec2)) \#* Y$ **and** $(p \cdot (yvec1@yvec2)) \#* Z$ **using** $\langle A_P \#* C' \rangle$
by $(\text{elim } cBrOpen(4)[\text{where } b=C \text{ and } ba=C' \text{ and } bd=V \text{ and } be=W \text{ and } bf=x\#y\#X \text{ and } bg=Y \text{ and } bh=Z])$ $(\text{assumption} \mid \text{simp})+$

from $\langle A_{P'} \#* (x\#y\#X) \rangle$ **have** $x \# A_{P'}$ **and** $y \# A_{P'}$ **and** $A_{P'} \#* X$ **by** $\text{simp}+$
from $\langle (p \cdot (yvec1@yvec2)) \#* (x\#y\#X) \rangle$ **have** $x \# (p \cdot (yvec1@yvec2))$ **and** $y \# (p \cdot (yvec1@yvec2))$ **and** $(p \cdot (yvec1@yvec2)) \#* X$ **by** $\text{simp}+$

from $\langle x \# \alpha \rangle$ $yvecEq$ **have** $x \# yvec1$ **and** $x \neq y$ **and** $x \# yvec2$ **by** $\text{simp}+$
from $\langle \text{distinct}(bn \alpha) \rangle$ $yvecEq$ **have** $yvec1 \#* yvec2$ **and** $y \# yvec1$ **and** $y \# yvec2$ **by** $\text{simp}+$
from $\langle bn \alpha \#* C' \rangle$ $yvecEq$ **have** $yvec1 \#* C'$ **and** $y \# C'$ **and** $yvec2 \#* C'$ **by** $\text{simp}+$

from $S \langle x \# \alpha \rangle \langle x \# p \cdot (yvec1@yvec2) \rangle$ $yvecEq$ **have** $x \# p$ **by** $(\text{intro freshAlphaSwap})$ $(\text{assumption} \mid \text{simp})+$
from $S \langle \text{distinct}(bn \alpha) \rangle \langle y \# p \cdot (yvec1@yvec2) \rangle$ $yvecEq$ **have** $y \# p$ **by** $(\text{intro freshAlphaSwap})$ $(\text{assumption} \mid \text{simp})+$

from $yvecEq S \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \cdot (yvec1@yvec2) \rangle \langle y \# p \cdot (yvec1@yvec2) \rangle$
have $\text{set } ((y, x)\#p) \subseteq \text{set}(bn \alpha) \times \text{set}(bn((y, x)\#p) \cdot \alpha)$
apply $(\text{simp add: bnEqvt[symmetric]})$
by $(\text{auto simp add: eqvts calc-atm})$

moreover from $PeqP'$ **have** $([(y, x)] \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq [(y, x)] \cdot \Psi_{P'}$
by $(\text{simp add: AssertionStatEqClosed})$
then have $([(y, x)\#p] \cdot \Psi_P) \otimes ([[(y, x)] \cdot \Psi']) \simeq ([[(y, x)] \cdot \Psi_{P'}])$
by (simp add: eqvts)
moreover from $\langle \text{distinctPerm } p \rangle S \langle x \neq y \rangle \langle x \# p \rangle \langle y \# p \rangle$ **have** $\text{distinctPerm}((y, x)\#p)$
by simp
moreover from FrP' **have** $([(x, y)] \cdot (\text{extractFrame}([(x, y)] \cdot P''))) = ((x, y)) \cdot \langle A_{P'}, \Psi_{P'} \rangle$
by simp
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $\text{extractFrame } P'' = \langle A_{P'}, ([[(y, x)] \cdot \Psi_{P'}]) \rangle$
by $(\text{simp add: eqvts name-swap})$

moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# N' \rangle \langle (p \cdot (yvec1@yvec2)) \#* [(x, y)] \cdot N' \rangle$ **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \#* [(y, x)] \cdot [(x, y)] \cdot N'$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
then have $((y, x)\#p) \cdot (yvec1@yvec2)) \#* N'$ **by** (*simp add: name-swap*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# C \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle x \# N' \rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \#* N'$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: eqvts perm-compose calc-atm freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# N' \rangle \langle (p \cdot (yvec1@yvec2)) \#* [(x, y)] \cdot P'' \rangle$ **have** $([(y, x)] \cdot p \cdot (yvec1@yvec2)) \#* [(y, x)] \cdot [(x, y)] \cdot P''$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
then have $((y, x)\#p) \cdot (yvec1@yvec2)) \#* P''$ **by** (*simp add: name-swap*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle \langle y \# p \rangle$
 $\langle x \# P'' \rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \#* P''$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# A_P' \rangle \langle (p \cdot (yvec1@yvec2)) \#* A_P' \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* A_P'$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* [(y, x)] \cdot A_P'$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# A_P' \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle y \# A_P' \rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \#* A_P'$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# C' \rangle \langle (p \cdot (yvec1@yvec2)) \#* C' \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* C'$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* [(y, x)] \cdot C'$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# C' \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle y \# C' \rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \#* C'$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# X \rangle \langle (p \cdot (yvec1@yvec2)) \#* X \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* X$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* [(y, x)] \cdot X$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# X \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle bn \alpha \#* X \rangle$ *yvecEq*
have $bn(((y, x)\#p) \cdot \alpha) \#* X$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# V \rangle \langle (p \cdot (yvec1@yvec2)) \#* V \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \#* V$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \#* [(y, x)] \cdot V$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# V \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$

$\langle y \# p \rangle \langle bn \ \alpha \ \#* \ V \rangle \ yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \ \#* \ V$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# W \rangle \langle (p \cdot (yvec1@yvec2)) \ \#* \ W \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \ \#* \ W$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \ \#* \ ([(y, x)] \cdot W)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# W \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle bn \ \alpha \ \#* \ W \rangle \ yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \ \#* \ W$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# Y \rangle \langle (p \cdot (yvec1@yvec2)) \ \#* \ Y \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \ \#* \ Y$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \ \#* \ ([(y, x)] \cdot Y)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# Y \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle bn \ \alpha \ \#* \ Y \rangle \ yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \ \#* \ Y$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle x \# p \rangle \langle y \# p \rangle \langle x \# Z \rangle \langle (p \cdot (yvec1@yvec2)) \ \#* \ Z \rangle$ **have** $(p \cdot (yvec1@x\#yvec2)) \ \#* \ Z$
by (*simp add: eqvts freshChainSimps*)
then have $([(y, x)] \cdot p \cdot (yvec1@x\#yvec2)) \ \#* \ ([(y, x)] \cdot Z)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \neq y \rangle \langle x \# Z \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle \langle x \# p \rangle$
 $\langle y \# p \rangle \langle bn \ \alpha \ \#* \ Z \rangle \ yvecEq$
have $bn(((y, x)\#p) \cdot \alpha) \ \#* \ Z$ **by** (*simp add: bnEqvt[symmetric]*) (*simp add: perm-compose calc-atm eqvts freshChainSimps*)
moreover from $\langle (yvec1@yvec2) \ \#* \ A_{P'} \rangle \langle y \# A_{P'} \rangle \ yvecEq$ **have** $bn \ \alpha \ \#* \ A_{P'}$
by *simp*
moreover from $\langle A_{P'} \ \#* \ ([(x, y)] \cdot N') \rangle$ **have** $([(x, y)] \cdot A_{P'}) \ \#* \ ([(x, y)] \cdot [(x, y)] \cdot N')$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $A_{P'} \ \#* \ N'$ **by** *simp*
with $\langle A_{P'} \ \#* \ M \rangle \langle (yvec1@yvec2) \ \#* \ A_{P'} \rangle \langle y \# A_{P'} \rangle \ \alpha eq$ **have** $A_{P'} \ \#* \ \alpha$ **by** *simp*
moreover then have $((y, x)\#p) \cdot A_{P'} \ \#* \ ((y, x)\#p) \cdot \alpha$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle \ S \ \langle (yvec1@yvec2) \ \#* \ A_{P'} \rangle \langle (p \cdot (yvec1@yvec2)) \ \#* \ A_{P'} \rangle$
have $A_{P'} \ \#* \ (((y, x)\#p) \cdot \alpha)$ **by** (*simp add: eqvts*)
moreover from $\langle A_{P'} \ \#* \ ([(x, y)] \cdot P') \rangle$ **have** $([(x, y)] \cdot A_{P'}) \ \#* \ ([(x, y)] \cdot [(x, y)] \cdot P')$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{P'} \rangle \langle y \# A_{P'} \rangle$ **have** $A_{P'} \ \#* \ P''$ **by** *simp*
moreover from $yvecEq \ \alpha eq \ \langle (p \cdot (yvec1@yvec2)) \ \#* \ (yvec1@yvec2) \rangle \langle y \# p \rangle$
 $\langle x \# \alpha \rangle \ S \ \langle (p \cdot (yvec1@yvec2)) \ \#* \ M \rangle \langle (p \cdot (yvec1@yvec2)) \ \#* \ ([(x, y)] \cdot N') \rangle \langle y \#$

```

yvec1 › ⟨y # yvec2 › ⟨x # p ›
  have bn(((y, x)#p) · α) #* α
    apply(simp add: eqvts del: set-append)
    apply(intro conjI)
      apply(simp add: perm-compose eqvts del: set-append)
      apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts
del: set-append)
        apply(simp add: perm-compose eqvts del: set-append)
        apply(simp add: perm-compose eqvts swapStarFresh del: set-append)
        apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
          apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
            apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
              apply(simp add: perm-compose freshChainSimps(6) swapStarFresh calc-atm
eqvts del: set-append)
                apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
                  apply(subst pt-fresh-star-bij[symmetric, OF pt-name-inst, OF at-name-inst,
where pi=[[x, y]])]
                    apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
                      apply(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del:
set-append)
                        apply(subst pt-fresh-star-bij[symmetric, OF pt-name-inst, OF at-name-inst,
where pi=[[x, y]])]
                          by(simp add: perm-compose freshChainSimps(6) calc-atm eqvts del: set-append)
                          moreover note ⟨AP' #* C⟩ ⟨AP' #* V⟩ ⟨AP' #* W⟩ ⟨AP' #* X⟩ ⟨AP' #* Y⟩
                          ⟨AP' #* Z⟩ ⟨distinct AP'⟩

      ultimately show ?case
        by(elim cBrOpen)
    next
      case(cScope Ψ P α P' x AP ΨP C C' α' P'' V W X Y Z)
      from ⟨α < (νx)P' = α' < P''⟩ ⟨x # α⟩ ⟨x # α'⟩
      obtain P''' where α < P' = α' < P''' and P'' = (νx)P'''
      apply(cases rule: actionCases[where α=α])
      apply(simp-all add: residualInject)
      apply (metis bn.simps(3) boundOutputScopeDest)
      by (metis bn.simps(4) boundOutputScopeDest)
      then obtain p Ψ' AP' ΨP' where S: set p ⊆ set(bn α') × set(bn(p · α'))
and PeqP': ((p · ΨP) ⊗ Ψ') ≈ ΨP'
      and distinctPerm p and bn(p · α') #* C' and FrP': extractFrame P''' =
⟨AP', ΨP'⟩
      and AP' #* P''' and AP' #* α' and AP' #* (p · α') and AP' #* C and
distinct AP'
      and bn(p · α') #* P''' and AP' #* V and AP' #* W and AP' #* (x#X)
and AP' #* Y and bn(p · α') #* α'

```

and $A_{P'} \#* Z$ **and** $bn(p \cdot \alpha') \#* V$ **and** $bn(p \cdot \alpha') \#* W$ **and** $bn(p \cdot \alpha') \#*$
 $(x\#X)$ **and** $bn(p \cdot \alpha') \#* Y$
and $bn(p \cdot \alpha') \#* Z$ **using** $cScope$
by($elim\ cScope$) ($assumption \mid simp$)+
from $\langle A_{P'} \#* (x\#X) \rangle$ **have** $x \# A_{P'}$ **and** $A_{P'} \#* X$ **by** $simp+$
from $\langle bn(p \cdot \alpha') \#* (x\#X) \rangle$ **have** $x \# bn(p \cdot \alpha')$ **and** $bn(p \cdot \alpha') \#* X$ **by**
 $simp+$

note $S\ PeqP' \langle distinctPerm\ p \rangle \langle bn(p \cdot \alpha') \#* C' \rangle$
moreover from $FrP' \langle P'' = (\nu x)P''' \rangle$ **have** $extractFrame\ P'' = \langle (x\#A_{P'})$,
 $\Psi_P \rangle$ **by** $simp$
moreover from $\langle A_{P'} \#* P''' \rangle \langle P'' = (\nu x)P''' \rangle \langle x \# A_{P'} \rangle$ **have** $(x\#A_{P'}) \#*$
 P'' **by**($simp\ add: abs-fresh$)
moreover from $\langle A_{P'} \#* \alpha' \rangle \langle A_{P'} \#* C \rangle \langle x \# \alpha' \rangle \langle x \# C \rangle$ **have** $(x\#A_{P'}) \#*$
 α' **and** $(x\#A_{P'}) \#* C$ **by** $simp+$
moreover note $\langle bn(p \cdot \alpha') \#* \alpha' \rangle$
moreover from $\langle bn(p \cdot \alpha') \#* P''' \rangle \langle P'' = (\nu x)P''' \rangle \langle x \# bn(p \cdot \alpha') \rangle$ **have**
 $bn(p \cdot \alpha') \#* P''$ **by** $simp$
moreover from $\langle A_{P'} \#* \alpha' \rangle \langle x \# \alpha' \rangle$ **have** $(x\#A_{P'}) \#* \alpha'$ **by** $simp$
moreover from $\langle A_{P'} \#* (p \cdot \alpha') \rangle \langle x \# \alpha' \rangle S \langle x \# bn(p \cdot \alpha') \rangle$ **have** $(x\#A_{P'})$
 $\#* (p \cdot \alpha')$
by($simp\ add: subjectEqvt[symmetric]\ bnEqvt[symmetric]\ objectEqvt[symmetric]$
 $freshChainSimps$)
moreover from $\langle A_{P'} \#* V \rangle \langle x \# V \rangle$ **have** $(x\#A_{P'}) \#* V$ **by** $simp+$
moreover from $\langle A_{P'} \#* W \rangle \langle x \# W \rangle$ **have** $(x\#A_{P'}) \#* W$ **by** $simp+$
moreover from $\langle A_{P'} \#* X \rangle \langle x \# X \rangle$ **have** $(x\#A_{P'}) \#* X$ **by** $simp+$
moreover from $\langle A_{P'} \#* Y \rangle \langle x \# Y \rangle$ **have** $(x\#A_{P'}) \#* Y$ **by** $simp+$
moreover from $\langle A_{P'} \#* Z \rangle \langle x \# Z \rangle$ **have** $(x\#A_{P'}) \#* Z$ **by** $simp+$
moreover note $\langle bn(p \cdot \alpha') \#* V \rangle \langle bn(p \cdot \alpha') \#* W \rangle \langle bn(p \cdot \alpha') \#* X \rangle \langle bn(p$
 $\cdot \alpha') \#* Y \rangle \langle bn(p \cdot \alpha') \#* Z \rangle$
moreover from $\langle distinct\ A_{P'} \rangle \langle x \# A_{P'} \rangle$ **have** $distinct(x\#A_{P'})$ **by** $simp$
ultimately show $?case$ **by**($elim\ cScope$)

next
case($cBang\ \Psi\ P\ A_P\ \Psi_P\ C\ C'\ \alpha\ P'\ V\ W\ X\ Y\ Z$)
then obtain $p\ \Psi'\ A_{P'}\ \Psi_{P'}$ **where** $S: set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$
and $FrP': extractFrame\ P' = \langle A_{P'}, \Psi_P \rangle$
and $PeqP': (p \cdot (\Psi_P \otimes \mathbf{1})) \otimes \Psi' \simeq \Psi_{P'}$
and $A_{P'} \#* C$ **and** $A_{P'} \#* P'$ **and** $A_{P'} \#* \alpha$ **and** $A_{P'} \#* (p \cdot \alpha)$
and $A_{P'} \#* V$ **and** $A_{P'} \#* W$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* Z$

and $distinct\ A_{P'}$
and $distinctPerm\ p$ **and** $(bn(p \cdot \alpha)) \#* \alpha$ **and** $(bn(p \cdot \alpha)) \#* P'$
and $(bn(p \cdot \alpha)) \#* C'$ **and** $(bn(p \cdot \alpha)) \#* V$ **and** $(bn(p \cdot \alpha)) \#* W$ **and**
 $(bn(p \cdot \alpha)) \#* X$ **and** $(bn(p \cdot \alpha)) \#* Y$ **and** $(bn(p \cdot \alpha)) \#* Z$
apply –
by($rule\ cBang$)($assumption \mid simp\ (no-asm-use)$)+
moreover from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $(p \cdot \Psi_P) \simeq (p \cdot \mathbf{1})$
by($simp\ add: AssertionStatEqClosed$)
then have $(p \cdot \Psi_P) \simeq \mathbf{1}$ **by**($simp\ add: permBottom$)
with $PeqP'$ **have** $(\mathbf{1} \otimes \Psi') \simeq \Psi_{P'}$

by(*simp add: eqvts permBottom*) (*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)
ultimately show *?case* **using** *cBang*
by (*metis permBottom*)
qed

with *A* **have** *?thesis* **by** *blast*
}
moreover have *bn* α $\#^*$ ($[]::'a$ list) **and** *bn* α $\#^*$ ($[]::('a$ action) list) **and** *bn* α $\#^*$ ($[]::name$ list) **and** *bn* α $\#^*$ ($[]::'b$ list) **and** *bn* α $\#^*$ ($[]::('a, 'b, 'c)$ psi list)
and *A_P* $\#^*$ ($[]::'a$ list) **and** *A_P* $\#^*$ ($[]::('a$ action) list) **and** *A_P* $\#^*$ ($[]::name$ list)
and *A_P* $\#^*$ ($[]::'b$ list) **and** *A_P* $\#^*$ ($[]::('a, 'b, 'c)$ psi list)
by *simp+*
ultimately show *?thesis* **by** *blast*
qed

lemma *expandTauFrame*:

fixes Ψ $:: 'b$
and *P* $:: ('a, 'b, 'c)$ psi
and *P'* $:: ('a, 'b, 'c)$ psi
and *A_P* $:: name$ list
and Ψ_P $:: 'b$
and *C* $:: 'f::fs-name$

assumes $\Psi \triangleright P \mapsto_{\tau} \prec P'$
and *extractFrame* *P* = $\langle A_P, \Psi_P \rangle$
and *distinct* *A_P*
and *A_P* $\#^*$ *P*
and *A_P* $\#^*$ *C*

obtains $\Psi' A_{P'} \Psi_{P'}$ **where** *extractFrame* *P'* = $\langle A_{P'}, \Psi_{P'} \rangle$ **and** $\Psi_P \otimes \Psi' \simeq \Psi_{P'}$
and *A_{P'}* $\#^*$ *C* **and** *A_{P'}* $\#^*$ *P'* **and** *distinct* *A_{P'}*

proof –

assume *A*: $\bigwedge A_{P'} \Psi_{P'} \Psi'$.
 $\llbracket extractFrame P' = \langle A_{P'}, \Psi_{P'} \rangle; \Psi_P \otimes \Psi' \simeq \Psi_{P'}; A_{P'} \#^* C; A_{P'} \#^* P';$
distinct A_{P'} \rrbracket
 $\implies thesis$

from $\langle \Psi \triangleright P \mapsto_{\tau} \prec P' \rangle \langle A_P \#^* P \rangle$ **have** *A_P* $\#^*$ *P'* **by**(*rule tauFreshChain-Derivative*)

{
fix *X* $:: name$ list
and *Y* $:: 'b$ list
and *Z* $:: ('a, 'b, 'c)$ psi list

assume *A_P* $\#^*$ *X*
and *A_P* $\#^*$ *Y*
and *A_P* $\#^*$ *Z*

with *assms* $\langle A_P \#* P' \rangle$ **obtain** $\Psi' A_P' \Psi_P'$ **where** *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$
 $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** $A_P' \#* C$
and $A_P' \#* P'$ **and** $A_P' \#* X$ **and** $A_P' \#* Y$ **and** $A_P' \#* Z$ **and** *distinct* A_P'
proof(*nominal-induct avoiding: C X Y Z arbitrary: thesis rule: tauFrameInduct*)
case(*cAlpha* $\Psi P P' A_P \Psi_P p C X Y Z$)
then obtain $\Psi' A_P' \Psi_P'$ **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and**
 $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** *distinct* A_P'
and $A_P' \#* C$ **and** $A_P' \#* P'$ **and** $A_P' \#* X$ **and** $A_P' \#* Y$ **and** $A_P' \#* Z$
by *metis*

have S : *set* $p \subseteq \text{set } A_P \times \text{set}(p \cdot A_P)$ **by** *fact*

from *FrP'* **have** $(p \cdot \text{extractFrame } P') = p \cdot \langle A_P', \Psi_P' \rangle$ **by** *simp*
with $\langle A_P \#* P' \rangle \langle (p \cdot A_P) \#* P' \rangle S$ **have** *extractFrame* $P' = \langle (p \cdot A_P'), (p \cdot \Psi_P') \rangle$ **by**(*simp add: eqvts*)
moreover from $\langle \Psi_P \otimes \Psi' \simeq \Psi_P' \rangle$ **have** $(p \cdot (\Psi_P \otimes \Psi')) \simeq (p \cdot \Psi_P')$ **by**(*rule AssertionStatEqClosed*)
then have $(p \cdot \Psi_P) \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_P')$ **by**(*simp add: eqvts*)
moreover from $\langle A_P' \#* C \rangle$ **have** $(p \cdot A_P') \#* (p \cdot C)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* C \rangle \langle (p \cdot A_P) \#* C \rangle S$ **have** $(p \cdot A_P') \#* C$ **by** *simp*
moreover from $\langle A_P' \#* P' \rangle$ **have** $(p \cdot A_P') \#* (p \cdot P')$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* P' \rangle \langle (p \cdot A_P) \#* P' \rangle S$ **have** $(p \cdot A_P') \#* P'$ **by** *simp*
moreover from $\langle A_P' \#* X \rangle$ **have** $(p \cdot A_P') \#* (p \cdot X)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* X \rangle \langle (p \cdot A_P) \#* X \rangle S$ **have** $(p \cdot A_P') \#* X$ **by** *simp*
moreover from $\langle A_P' \#* Y \rangle$ **have** $(p \cdot A_P') \#* (p \cdot Y)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* Y \rangle \langle (p \cdot A_P) \#* Y \rangle S$ **have** $(p \cdot A_P') \#* Y$ **by** *simp*
moreover from $\langle A_P' \#* Z \rangle$ **have** $(p \cdot A_P') \#* (p \cdot Z)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* Z \rangle \langle (p \cdot A_P) \#* Z \rangle S$ **have** $(p \cdot A_P') \#* Z$ **by** *simp*
moreover from $\langle \text{distinct } A_P' \rangle$ **have** *distinct*($p \cdot A_P'$) **by** *simp*
ultimately show *?case* **by**(*rule cAlpha*)

next
case(*cCase* $\Psi P P' \varphi Cs A_P \Psi_P C B Y Z$ *thesis*)
then obtain $\Psi' A_P' \Psi_P'$ **where** *FrP'*: *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$
and $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** *distinct* A_P'
and $A_P' \#* C$ **and** $A_P' \#* P'$
and $A_P' \#* B$ **and** $A_P' \#* Y$ **and** $A_P' \#* Z$
apply –
by(*rule cCase*) (*assumption* | *simp (no-asm-use)*) +
with $\langle \Psi_P \simeq \mathbf{1} \rangle \langle \Psi_P \otimes \Psi' \simeq \Psi_P' \rangle$ **have** $\mathbf{1} \otimes \Psi' \simeq \Psi_P'$
by(*metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym*)
then show *?case* **using** *FrP'* $\langle A_P' \#* P' \rangle \langle A_P' \#* C \rangle \langle A_P' \#* B \rangle \langle A_P' \#* Y \rangle \langle A_P' \#* Z \rangle \langle \text{distinct } A_P' \rangle$ **using** *cCase*

by force
 next
 case(*cPar1* $\Psi \Psi_Q P P' A_Q Q A_P \Psi_P C X Y Z$)
 moreover from $\langle A_P \#* X \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* Y \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* Q \rangle$
 $\langle A_P \#* Z \rangle$
 have $A_P \#* (X @ A_Q)$ and $A_P \#* (\Psi_Q \# Y)$ and $A_P \#* (Q \# Z)$
 by *simp+*
 ultimately obtain $\Psi' A_P' \Psi_{P'}$ where *FrP'*: $\text{extractFrame } P' = \langle A_P', \Psi_{P'} \rangle$
 and *distinct* A_P'
 and $\Psi_P \otimes \Psi' \simeq \Psi_{P'}$ and $A_P \#* P$ and $A_P \#* C$ and $A_P' \#* C$ and $A_P' \#* P'$
 and $A_P' \#* (X @ A_Q)$ and $A_P' \#* (\Psi_Q \# Y)$ and $A_P' \#* (Q \# Z)$
 by *metis*

 then have $A_P' \#* X$ and $A_P' \#* A_Q$ and $A_P' \#* Y$ and $A_P' \#* \Psi_Q$ and
 $A_P' \#* Q$ and $A_P' \#* Z$
 by *simp+*

 from $\langle A_P' \#* A_Q \rangle \langle A_Q \#* P' \rangle$ *FrP'* have $A_Q \#* \Psi_{P'}$ by (*force dest: extract-FrameFreshChain*)
 with $\langle A_P' \#* \Psi_Q \rangle \langle A_P' \#* A_Q \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ *FrP'*
 have $\text{extractFrame}(P' \parallel Q) = \langle (A_P' @ A_Q), \Psi_{P'} \otimes \Psi_Q \rangle$ by *simp*

 moreover from $\langle \Psi_P \otimes \Psi' \simeq \Psi_{P'} \rangle$ have $(\Psi_P \otimes \Psi_Q) \otimes \Psi' \simeq \Psi_{P'} \otimes \Psi_Q$
 by (*metis Associativity Commutativity Composition AssertionStatEqTrans AssertionStatEqSym*)

 moreover from $\langle A_P' \#* C \rangle \langle A_Q \#* C \rangle$ have $(A_P' @ A_Q) \#* C$ by *simp*
 moreover from $\langle A_P' \#* P' \rangle \langle A_Q \#* P' \rangle \langle A_P' \#* Q \rangle \langle A_Q \#* Q \rangle$ have $(A_P' @ A_Q) \#* (P' \parallel Q)$ by *simp*
 moreover from $\langle A_P' \#* X \rangle \langle A_Q \#* X \rangle$ have $(A_P' @ A_Q) \#* X$ by *simp*
 moreover from $\langle A_P' \#* Y \rangle \langle A_Q \#* Y \rangle$ have $(A_P' @ A_Q) \#* Y$ by *simp*
 moreover from $\langle A_P' \#* Z \rangle \langle A_Q \#* Z \rangle$ have $(A_P' @ A_Q) \#* Z$ by *simp*
 moreover from $\langle A_P' \#* A_Q \rangle \langle \text{distinct } A_P' \rangle \langle \text{distinct } A_Q \rangle$ have *distinct* $(A_P' @ A_Q)$
 by *simp*
 ultimately show ?*case* by (*rule cPar1*)
 next
 case(*cPar2* $\Psi \Psi_P Q Q' A_P P A_Q \Psi_Q C X Y Z$)
 moreover from $\langle A_Q \#* X \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* Y \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* P \rangle$
 $\langle A_Q \#* Z \rangle$
 have $A_Q \#* (X @ A_P)$ and $A_Q \#* (\Psi_P \# Y)$ and $A_Q \#* (P \# Z)$
 by (*simp add: freshChainSimps*)
 ultimately obtain $\Psi' A_Q' \Psi_{Q'}$ where *FrQ'*: $\text{extractFrame } Q' = \langle A_Q', \Psi_{Q'} \rangle$
 and *distinct* A_Q'
 and $\Psi_Q \otimes \Psi' \simeq \Psi_{Q'}$ and $A_Q' \#* C$ and $A_Q' \#* Q'$
 and $A_Q' \#* (X @ A_P)$ and $A_Q' \#* (\Psi_P \# Y)$ and $A_Q' \#* (P \# Z)$
 by *metis*

 then have $A_Q' \#* X$ and $A_Q' \#* A_P$ and $A_Q' \#* Y$ and $A_Q' \#* \Psi_P$ and

$A_{Q'} \#* P$ and $A_{Q'} \#* Z$
by simp+

from $\langle A_{Q'} \#* A_P \rangle \langle A_P \#* Q' \rangle \text{Fr}Q'$ **have** $A_P \#* \Psi_{Q'}$ **by** (*force dest: extractFrameFreshChain*)

with $\langle A_{Q'} \#* \Psi_P \rangle \langle A_{Q'} \#* A_P \rangle \langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \text{Fr}Q'$
have $\text{extractFrame}(P \parallel Q') = \langle (A_P @ A_{Q'}), \Psi_P \otimes \Psi_{Q'} \rangle$ **by simp**

moreover from $\langle \Psi_Q \otimes \Psi' \simeq \Psi_{Q'} \rangle$ **have** $(\Psi_P \otimes \Psi_Q) \otimes \Psi' \simeq \Psi_P \otimes \Psi_{Q'}$
by (*metis Associativity Commutativity Composition AssertionStatEqTrans AssertionStatEqSym*)

moreover from $\langle A_P \#* C \rangle \langle A_{Q'} \#* C \rangle$ **have** $(A_P @ A_{Q'}) \#* C$ **by simp**
moreover from $\langle A_P \#* P \rangle \langle A_{Q'} \#* P \rangle \langle A_P \#* Q' \rangle \langle A_{Q'} \#* Q' \rangle$ **have** $(A_P @ A_{Q'}) \#* (P \parallel Q')$ **by simp**

moreover from $\langle A_P \#* X \rangle \langle A_{Q'} \#* X \rangle$ **have** $(A_P @ A_{Q'}) \#* X$ **by simp**
moreover from $\langle A_P \#* Y \rangle \langle A_{Q'} \#* Y \rangle$ **have** $(A_P @ A_{Q'}) \#* Y$ **by simp**
moreover from $\langle A_P \#* Z \rangle \langle A_{Q'} \#* Z \rangle$ **have** $(A_P @ A_{Q'}) \#* Z$ **by simp**
moreover from $\langle A_{Q'} \#* A_P \rangle \langle \text{distinct } A_P \rangle \langle \text{distinct } A_{Q'} \rangle$ **have** $\text{distinct}(A_P @ A_{Q'})$
by simp

ultimately show $?case$ **by** (*rule cPar2*)

next

case (*cComm1* $\Psi \Psi_Q P M N P' A_P \Psi_P Q K \text{vec } Q' A_Q C X Y Z$)
have $P\text{Trans}: \Psi \otimes \Psi_Q \triangleright P \mapsto M(|N|) \prec P'$ **and** $Q\text{Trans}: \Psi \otimes \Psi_P \triangleright Q \mapsto K(|\nu * \text{vec}|) \langle N \rangle \prec Q'$ **by fact+**

from $P\text{Trans}$ $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle \text{distinct } A_P \rangle \langle A_P \#* P \rangle \langle A_P \#* N \rangle \langle A_P \#* M \rangle$
 $\langle A_P \#* Q' \rangle \langle A_P \#* C \rangle \langle A_P \#* X \rangle \langle A_P \#* Y \rangle \langle A_P \#* Z \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \text{vec} \rangle$

obtain $\Psi' A_{P'} \Psi_{P'}$ **where** $\text{Fr}P': \text{extractFrame } P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** $\text{Peq}P': \Psi_P \otimes \Psi' \simeq \Psi_{P'}$

and $A_{P'} \#* Q'$ **and** $A_{P'} \#* C$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $\text{distinct } A_{P'}$

and $A_{P'} \#* Z$ **and** $A_{P'} \#* A_Q$ **and** $A_{P'} \#* \text{vec}$ **and** $A_{P'} \#* P'$

by (*elim expandNonTauFrame* [**where** $C = (Q', C, X, Y, Z, A_Q, \text{vec})$ **and** $C' = (Q', C, X, Y, Z, A_Q, \text{vec})$]) *auto*

moreover from $Q\text{Trans}$ **have** $\text{distinct } \text{vec}$ **by** (*auto dest: boundOutputDistinct*)

from $Q\text{Trans}$ $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle \text{distinct } A_Q \rangle \langle A_Q \#* P \rangle \langle A_Q \#* \text{vec} \rangle \langle A_Q \#* K \rangle \langle \text{vec} \#* K \rangle \langle \text{distinct } \text{vec} \rangle$
 $\langle A_{P'} \#* A_Q \rangle \langle A_{P'} \#* \text{vec} \rangle \langle A_Q \#* Q \rangle \langle \text{vec} \#* Q \rangle \langle A_Q \#* \Psi_P \rangle \langle \text{vec} \#* \Psi_P \rangle \langle A_Q \#* N \rangle$
 $\langle A_Q \#* C \rangle \langle A_Q \#* X \rangle \langle A_Q \#* Y \rangle \langle A_Q \#* Z \rangle \langle \text{vec} \#* P \rangle \langle \text{vec} \#* C \rangle \langle \text{vec} \#* X \rangle \langle \text{vec} \#* Y \rangle \langle \text{vec} \#* Z \rangle$

obtain $p \Psi'' A_{Q'} \Psi_{Q'}$ **where** $S: \text{set } p \subseteq \text{set } \text{vec} \times \text{set}(p \cdot \text{vec})$ **and** $\text{Qeq}Q': (p \cdot \Psi_Q) \otimes \Psi'' \simeq \Psi_{Q'}$ **and** $\text{Fr}Q': \text{extractFrame } Q' = \langle A_{Q'}, \Psi_{Q'} \rangle$

and $A_{Q'} \#* A_{P'}$ **and** $A_{Q'} \#* C$ **and** $A_{Q'} \#* X$ **and** $A_{Q'} \#* Y$ **and** $A_{Q'} \#* Z$ **and** $A_{Q'} \#* P$ **and** $A_{Q'} \#* N$ **and** $\text{distinct } A_{Q'}$

and $(p \cdot \text{vec}) \#* A_{P'}$ **and** $(p \cdot \text{vec}) \#* C$ **and** $(p \cdot \text{vec}) \#* X$ **and** $(p \cdot \text{vec}) \#* Y$ **and** $(p \cdot \text{vec}) \#* P$

and $(p \cdot xvec) \#* Z$ **and** $(p \cdot xvec) \#* N$ **and** $(p \cdot xvec) \#* \Psi_P$ **and** $(p \cdot xvec) \#* A_Q'$ **and** $(p \cdot xvec) \#* Q'$
and *distinctPerm* p **and** $A_Q' \#* xvec$ **and** $A_Q' \#* Q'$
by (*elim expandNonTauFrame*[**where** $C=(P, C, X, Y, Z, A_P', \Psi_P)$ **and** $C'=(P, C, X, Y, Z, A_P', \Psi_P)$]) (*assumption* | *simp*)**+**

from $PTrans \langle A_Q' \#* P \rangle \langle A_Q' \#* N \rangle \langle (p \cdot xvec) \#* P \rangle \langle (p \cdot xvec) \#* N \rangle$
have $A_Q' \#* P'$ **and** $(p \cdot xvec) \#* P'$ **by** (*force dest: inputFreshChainDerivative*)**+**
with $FrP' \langle A_Q' \#* A_P' \rangle \langle (p \cdot xvec) \#* A_P' \rangle$ **have** $A_Q' \#* \Psi_{P'}$ **and** $(p \cdot xvec) \#* \Psi_{P'}$ **by** (*force dest: extractFrameFreshChain*)**+**
from $FrQ' \langle A_Q' \#* A_P' \rangle \langle A_P' \#* Q' \rangle \langle (p \cdot xvec) \#* A_Q' \rangle \langle (p \cdot xvec) \#* Q' \rangle$
have $A_P' \#* \Psi_{Q'}$ **and** $(p \cdot xvec) \#* \Psi_{Q'}$
by (*force dest: extractFrameFreshChain*)**+**

have $extractFrame(\nu * xvec)(P' \parallel Q') = \langle ((p \cdot xvec) @_{A_P'} @_{A_Q'}), (p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_{Q'}) \rangle$
proof –
from $FrP' FrQ' \langle A_P' \#* \Psi_{Q'} \rangle \langle A_Q' \#* A_P' \rangle \langle A_Q' \#* \Psi_{P'} \rangle$ **have** $extractFrame(P' \parallel Q') = \langle (A_P' @_{A_Q'}), \Psi_{P'} \otimes \Psi_{Q'} \rangle$
by *simp*
then have $extractFrame(\nu * xvec)(P' \parallel Q') = \langle (xvec @_{A_P'} @_{A_Q'}), \Psi_{P'} \otimes \Psi_{Q'} \rangle$
by (*induct xvec*) *auto*
moreover from $\langle (p \cdot xvec) \#* \Psi_{P'} \rangle \langle (p \cdot xvec) \#* \Psi_{Q'} \rangle S$
have $(\nu * xvec)((\nu * (A_P' @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_{Q'}))) = (\nu * (p \cdot xvec))(p \cdot (\nu * (A_P' @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_{Q'})))$
by (*intro frameChainAlpha*) (*auto simp add: fresh-star-def frameResChainFresh*)
then have $(\nu * xvec)((\nu * (A_P' @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_{Q'}))) = (\nu * (p \cdot xvec))((\nu * (A_P' @_{A_Q'}))(FAssert((p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_{Q'}))))$
using $\langle A_P' \#* xvec \rangle \langle (p \cdot xvec) \#* A_P' \rangle \langle A_Q' \#* xvec \rangle \langle (p \cdot xvec) \#* A_Q' \rangle S$
by (*auto simp add: eqvts*)
ultimately show *?thesis*
by (*simp add: frameChainAppend*)
qed

moreover have $(\Psi_P \otimes \Psi_Q) \otimes ((p \cdot \Psi') \otimes (p \cdot \Psi'')) \simeq (p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_{Q'})$
proof –
have $(\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi'') \simeq (\Psi_P \otimes \Psi') \otimes ((p \cdot \Psi_Q) \otimes \Psi'')$
by (*metis Associativity Commutativity Composition AssertionStatEqTrans*)
moreover from $PeqP' QeqQ'$ **have** $(\Psi_P \otimes \Psi') \otimes ((p \cdot \Psi_Q) \otimes \Psi'') \simeq \Psi_{P'} \otimes \Psi_{Q'}$
by (*metis Associativity Commutativity Composition AssertionStatEqTrans*)
ultimately have $(\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi'') \simeq \Psi_{P'} \otimes \Psi_{Q'}$
by (*metis AssertionStatEqTrans*)
then have $(p \cdot ((\Psi_P \otimes (p \cdot \Psi_Q)) \otimes (\Psi' \otimes \Psi''))) \simeq (p \cdot (\Psi_{P'} \otimes \Psi_{Q'}))$
by (*rule AssertionStatEqClosed*)
with $\langle xvec \#* \Psi_P \rangle \langle (p \cdot xvec) \#* \Psi_P \rangle S \langle distinctPerm p \rangle$ **show** *?thesis*

by(*simp add: eqvts*)
qed

moreover from $\langle (p \cdot xvec) \#* C \rangle \langle A_{P'} \#* C \rangle \langle A_{Q'} \#* C \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* C$ **by** *simp*
moreover from $\langle (p \cdot xvec) \#* X \rangle \langle A_{P'} \#* X \rangle \langle A_{Q'} \#* X \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* X$ **by** *simp*
moreover from $\langle (p \cdot xvec) \#* Y \rangle \langle A_{P'} \#* Y \rangle \langle A_{Q'} \#* Y \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* Y$ **by** *simp*
moreover from $\langle (p \cdot xvec) \#* Z \rangle \langle A_{P'} \#* Z \rangle \langle A_{Q'} \#* Z \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* Z$ **by** *simp*
moreover from $\langle (p \cdot xvec) \#* P' \rangle \langle (p \cdot xvec) \#* Q' \rangle \langle A_{P'} \#* P' \rangle \langle A_{P'} \#* Q' \rangle \langle A_{Q'} \#* P' \rangle \langle A_{Q'} \#* Q' \rangle$
have $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* ((\nu * xvec)(P' \parallel Q'))$ **by**(*auto simp add: resChainFresh fresh-star-def*)
moreover from $\langle (p \cdot xvec) \#* A_{P'} \rangle \langle (p \cdot xvec) \#* A_{Q'} \rangle \langle A_{Q'} \#* A_{P'} \rangle \langle distinct\ xvec \rangle \langle distinct\ A_{P'} \rangle \langle distinct\ A_{Q'} \rangle$
have *distinct* $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}})$
by *auto (simp add: name-list-supp fresh-star-def fresh-def)+*

ultimately show *?case using cComm1*
by *metis*
next

case(*cComm2* $\Psi \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q C X Y Z$)
have *PTrans*: $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)(N) \prec P'$ **and** *QTrans*: $\Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'$ **by** *fact+*
from *PTrans* **have** *distinct xvec* **by**(*auto dest: boundOutputDistinct*)
from *PTrans* $\langle extractFrame\ P = \langle A_P, \Psi_P \rangle \rangle \langle distinct\ A_P \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle xvec \#* Q \rangle \langle distinct\ xvec \rangle \langle xvec \#* M \rangle$
 $\langle A_P \#* C \rangle \langle A_P \#* X \rangle \langle A_P \#* Y \rangle \langle A_P \#* Z \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* xvec \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$
 $\langle xvec \#* P \rangle \langle xvec \#* C \rangle \langle xvec \#* X \rangle \langle xvec \#* Y \rangle \langle xvec \#* Z \rangle \langle A_Q \#* xvec \rangle \langle xvec \#* \Psi_Q \rangle$
obtain $p \Psi' A_{P'} \Psi_{P'}$ **where** $S: set\ p \subseteq set\ xvec \times set(p \cdot xvec)$
and *FrP'*: $extractFrame\ P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** *PeqP'*: $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$
and *distinct A_{P'}*
and $A_{P'} \#* C$ **and** $A_{P'} \#* X$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* N$ **and** $A_{P'} \#* Q$
and $(p \cdot xvec) \#* Q$
and $A_{P'} \#* Z$ **and** $A_{P'} \#* A_Q$ **and** $A_{P'} \#* xvec$ **and** $A_{P'} \#* P'$ **and** $(p \cdot xvec) \#* N$ **and** $(p \cdot xvec) \#* \Psi_Q$
and $(p \cdot xvec) \#* A_{P'}$ **and** $(p \cdot xvec) \#* C$ **and** $(p \cdot xvec) \#* X$ **and** $(p \cdot xvec) \#* A_Q$
and $(p \cdot xvec) \#* Y$ **and** $(p \cdot xvec) \#* Z$ **and** $(p \cdot xvec) \#* P'$ **and** *distinctPerm p*
by(*elim expandNonTauFrame[where C=(C, X, Y, Z, A_Q, Q, Ψ_Q) and C'=(C, X, Y, Z, A_Q, Q, Ψ_Q)] (assumption | simp)+*)

from *QTrans* $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle \langle distinct\ A_Q \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* P' \rangle \langle (p \cdot xvec) \#* A_Q \rangle$

$\langle A_{P'} \#* A_Q \rangle \langle A_Q \#* P \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* C \rangle \langle A_Q \#* X \rangle \langle A_Q \#* Y \rangle \langle A_Q \#* Z \rangle \langle A_Q \#* N \rangle \langle A_Q \#* K \rangle$
obtain $\Psi'' A_Q' \Psi_Q'$ **where** $QeqQ': \Psi_Q \otimes \Psi'' \simeq \Psi_Q'$ **and** $FrQ': extractFrame Q' = \langle A_Q', \Psi_Q' \rangle$ **and** *distinct* A_Q'
and $A_Q' \#* xvec$ **and** $A_Q' \#* Q'$ **and** $A_Q' \#* xvec$ **and** $A_Q' \#* P'$ **and** $A_Q' \#* (p \cdot xvec)$
and $A_Q' \#* A_{P'}$ **and** $A_Q' \#* C$ **and** $A_Q' \#* X$ **and** $A_Q' \#* Y$ **and** $A_Q' \#* Z$ **and** $A_Q' \#* P$
by(*elim expandNonTauFrame*[**where** $C=(P, C, P', X, Y, Z, A_{P'}, xvec, (p \cdot xvec), \Psi_P)$ **and** $C'=(P, C, P', X, Y, Z, A_{P'}, xvec, (p \cdot xvec), \Psi_P)$]) (*assumption* | *simp*)+

from $QTrans \langle A_{P'} \#* Q \rangle \langle A_{P'} \#* N \rangle \langle (p \cdot xvec) \#* Q \rangle \langle (p \cdot xvec) \#* N \rangle$
have $A_{P'} \#* Q'$ **and** $(p \cdot xvec) \#* Q'$ **by**(*force dest: inputFreshChainDerivative*)+
with $FrQ' \langle A_Q' \#* A_{P'} \rangle \langle A_Q' \#* (p \cdot xvec) \rangle$ **have** $A_{P'} \#* \Psi_Q'$ **and** $(p \cdot xvec) \#* \Psi_Q'$ **by**(*force dest: extractFrameFreshChain*)+
from $FrP' \langle A_Q' \#* A_{P'} \rangle \langle A_Q' \#* P' \rangle \langle (p \cdot xvec) \#* A_{P'} \rangle \langle (p \cdot xvec) \#* P' \rangle$
have $A_Q' \#* \Psi_{P'}$ **and** $(p \cdot xvec) \#* \Psi_{P'}$
by(*force dest: extractFrameFreshChain*)+

have $extractFrame((\nu*xvec))(P' \parallel Q') = \langle ((p \cdot xvec) @_{A_{P'}} @_{A_Q'}), (p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_Q') \rangle$
proof –
from $FrP' FrQ' \langle A_{P'} \#* \Psi_Q' \rangle \langle A_Q' \#* A_{P'} \rangle \langle A_Q' \#* \Psi_{P'} \rangle$ **have** $extractFrame(P' \parallel Q') = \langle (A_{P'} @_{A_Q'}), \Psi_{P'} \otimes \Psi_Q' \rangle$
by *simp*
then have $extractFrame((\nu*xvec))(P' \parallel Q') = \langle (xvec @_{A_{P'}} @_{A_Q'}), \Psi_{P'} \otimes \Psi_Q' \rangle$
by(*induct xvec*) *auto*
moreover from $\langle (p \cdot xvec) \#* \Psi_{P'} \rangle \langle (p \cdot xvec) \#* \Psi_Q' \rangle S$
have $(\nu*xvec)((\nu*(A_{P'} @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_Q')) = (\nu*(p \cdot xvec))(p \cdot (\nu*(A_{P'} @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_Q')))$
by(*intro frameChainAlpha*) (*auto simp add: fresh-star-def frameResChainFresh*)
then have $(\nu*xvec)((\nu*(A_{P'} @_{A_Q'}))(FAssert(\Psi_{P'} \otimes \Psi_Q')) = (\nu*(p \cdot xvec))((\nu*(A_{P'} @_{A_Q'}))(FAssert((p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_Q'))))$
using $\langle A_{P'} \#* xvec \rangle \langle (p \cdot xvec) \#* A_{P'} \rangle \langle A_Q' \#* xvec \rangle \langle A_Q' \#* (p \cdot xvec) \rangle S$
by(*auto simp add: eqvts*)
ultimately show *?thesis*
by(*simp add: frameChainAppend*)
qed

moreover have $(\Psi_P \otimes \Psi_Q) \otimes ((p \cdot \Psi') \otimes (p \cdot \Psi'')) \simeq (p \cdot \Psi_{P'}) \otimes (p \cdot \Psi_Q')$
proof –
have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'') \simeq ((p \cdot \Psi_P) \otimes \Psi') \otimes (\Psi_Q \otimes \Psi'')$
by(*metis Associativity Commutativity Composition AssertionStatEqTrans*)
moreover from $PeqP' QeqQ'$ **have** $((p \cdot \Psi_P) \otimes \Psi') \otimes (\Psi_Q \otimes \Psi'') \simeq \Psi_{P'} \otimes \Psi_Q'$

by(metis *Associativity Commutativity Composition AssertionStatEqTrans*)
ultimately have $((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'') \simeq \Psi_{P'} \otimes \Psi_{Q'}$
by(metis *AssertionStatEqTrans*)
then have $(p \cdot ((p \cdot \Psi_P) \otimes \Psi_Q) \otimes (\Psi' \otimes \Psi'')) \simeq (p \cdot (\Psi_{P'} \otimes \Psi_{Q'}))$
by(rule *AssertionStatEqClosed*)
with $\langle xvec \#* \Psi_Q \rangle \langle (p \cdot xvec) \#* \Psi_Q \rangle S \langle distinctPerm p \rangle$ **show** ?thesis
by(simp add: eqvts)
qed

moreover from $\langle (p \cdot xvec) \#* C \rangle \langle A_{P'} \#* C \rangle \langle A_{Q'} \#* C \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* C$ **by** simp
moreover from $\langle (p \cdot xvec) \#* X \rangle \langle A_{P'} \#* X \rangle \langle A_{Q'} \#* X \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* X$ **by** simp
moreover from $\langle (p \cdot xvec) \#* Y \rangle \langle A_{P'} \#* Y \rangle \langle A_{Q'} \#* Y \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* Y$ **by** simp
moreover from $\langle (p \cdot xvec) \#* Z \rangle \langle A_{P'} \#* Z \rangle \langle A_{Q'} \#* Z \rangle$ **have** $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* Z$ **by** simp
moreover from $\langle (p \cdot xvec) \#* P' \rangle \langle (p \cdot xvec) \#* Q' \rangle \langle A_{P'} \#* P' \rangle \langle A_{P'} \#* Q' \rangle \langle A_{Q'} \#* P' \rangle \langle A_{Q'} \#* Q' \rangle$
have $((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}}) \#* ((\nu * xvec)(P' \parallel Q'))$ **by**(auto simp add: resChainFresh fresh-star-def)
moreover from $\langle (p \cdot xvec) \#* A_{P'} \rangle \langle A_{Q'} \#* (p \cdot xvec) \rangle \langle A_{Q'} \#* A_{P'} \rangle \langle distinct xvec \rangle \langle distinct A_{P'} \rangle \langle distinct A_{Q'} \rangle$
have $distinct((p \cdot xvec) @_{A_{P'}} @_{A_{Q'}})$
by auto (simp add: name-list-supp fresh-star-def fresh-def)+

ultimately show ?case using cComm2 **by** metis
next

case(cBrClose $\Psi P M xvec N P' A_P \Psi_P x C X Y Z$)
from $\langle x \# A_P \rangle \#* C$ **have** $A_P \#* C$ **by** simp
from $\langle x \# A_P \rangle \#* X \rangle \langle x \# A_P \rangle$ **have** $A_P \#* (x \# X)$ **by** simp
from $\langle x \# A_P \rangle \#* Y$ **have** $A_P \#* Y$ **by** simp
from $\langle x \# A_P \rangle \#* Z$ **have** $A_P \#* Z$ **by** simp
from $\langle x \# xvec \rangle \langle xvec \#* X \rangle$ **have** $xvec \#* (x \# X)$ **by** simp

have $PTrans: \Psi \triangleright P \mapsto_{\downarrow} M(\nu * xvec) \langle N \rangle \prec P'$ **by** fact+
from $PTrans$ **have** $distinct xvec$ **by**(auto dest: boundOutputDistinct)
from $PTrans$ $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle distinct A_P \rangle \langle xvec \#* M \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* N \rangle \langle A_P \#* C \rangle$
 $\langle A_P \#* (x \# X) \rangle \langle A_P \#* Y \rangle \langle A_P \#* Z \rangle \langle xvec \#* P \rangle \langle xvec \#* C \rangle \langle xvec \#* (x \# X) \rangle \langle xvec \#* Y \rangle \langle xvec \#* Z \rangle$

obtain $p \Psi' A_{P'} \Psi_{P'}$ **where** $S: set p \subseteq set xvec \times set(p \cdot xvec)$
and $FrP': extractFrame P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** $PeqP': (p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$
and $distinct A_{P'}$
and $A_{P'} \#* C$ **and** $A_{P'} \#* (x \# X)$ **and** $A_{P'} \#* Y$ **and** $A_{P'} \#* N$
and $A_{P'} \#* Z$ **and** $A_{P'} \#* xvec$ **and** $A_{P'} \#* P'$ **and** $(p \cdot xvec) \#* xvec$ **and**
 $(p \cdot xvec) \#* N$
and $(p \cdot xvec) \#* A_{P'}$ **and** $(p \cdot xvec) \#* C$ **and** $(p \cdot xvec) \#* (x \# X)$

and $(p \cdot xvec) \#* Y$ **and** $(p \cdot xvec) \#* Z$ **and** $(p \cdot xvec) \#* P'$ **and** $distinctPerm$
 p
by $(elim\ expandNonTauFrame[where\ C=(C, (x\#X), Y, Z)$ **and** $C'=(C, (x\#X), Y, Z))$ $(assumption\ |\ simp)+$
then have $A_{P'} \#* X$ **and** $x \# A_{P'}$ **and** $(p \cdot xvec) \#* X$ **and** $x \# (p \cdot xvec)$ **by**
 $simp+$
from $FrP' \langle (p \cdot xvec) \#* A_{P'} \rangle \langle (p \cdot xvec) \#* P' \rangle$ **have** $(p \cdot xvec) \#* \Psi_{P'}$
by $(metis\ extractFrameFreshChain\ freshFrameDest)$
from $S \langle A_{P'} \#* xvec \rangle \langle (p \cdot xvec) \#* A_{P'} \rangle$ **have** $p \cdot A_{P'} = A_{P'}$ **by** $simp$

from $\langle (p \cdot xvec) \#* \Psi_{P'} \rangle S$
have $(\nu * xvec)(\langle A_{P'} \rangle, \Psi_{P'}) = (\nu * (p \cdot xvec))(p \cdot \langle A_{P'} \rangle, \Psi_{P'})$
by $(intro\ frameChainAlpha)$ $(auto\ simp\ add:\ fresh-star-def\ frameResChain-$
 $Fresh)$
then have $\langle (xvec @ A_{P'}) \rangle, \Psi_{P'} = (p \cdot \langle (xvec @ A_{P'}) \rangle, \Psi_{P'})$
by $(metis\ frameChainAppend\ frameResChainEqvt)$
with $\langle p \cdot A_{P'} = A_{P'} \rangle$ **have** $\langle (xvec @ A_{P'}) \rangle, \Psi_{P'} = \langle (p \cdot xvec) @ A_{P'} \rangle, (p \cdot$
 $\Psi_{P'})$
by $(simp\ add:\ eqvts)$
then have $\langle (x\#(xvec @ A_{P'})) \rangle, \Psi_{P'} = \langle (x\#((p \cdot xvec) @ A_{P'})) \rangle, (p \cdot \Psi_{P'})$
by $simp$

moreover from FrP' **have** $extractFrame ((\nu x)(\nu * xvec)P') = \langle (x\#(xvec @ A_{P'})) \rangle,$
 $\Psi_{P'}$
by $(metis\ extractFrameResChain\ extractFrame-extractFrame'-extractFrame''.simps$
 $frameChainAppend\ frameResChain.step)$

ultimately have $FrP'2:$ $extractFrame ((\nu x)(\nu * xvec)P') = \langle (x\#((p \cdot$
 $xvec) @ A_{P'})) \rangle, (p \cdot \Psi_{P'})$
by $simp$

from $PeqP'$ **have** $(p \cdot ((p \cdot \Psi_P) \otimes \Psi')) \simeq (p \cdot \Psi_{P'})$
by $(metis\ AssertionStatEqClosed)$
with $\langle distinctPerm\ p \rangle$ **have** $PeqP'2:$ $\Psi_P \otimes (p \cdot \Psi') \simeq (p \cdot \Psi_{P'})$
by $(simp\ add:\ eqvts)$

note $FrP'2\ PeqP'2$

moreover from $\langle x \# C \rangle \langle (p \cdot xvec) \#* C \rangle \langle A_{P'} \#* C \rangle$
have $(x\#((p \cdot xvec) @ A_{P'})) \#* C$ **by** $simp$
moreover from $\langle (x \# A_P) \#* (\nu x)(\nu * xvec)P' \rangle \langle A_{P'} \#* (x\#X) \rangle \langle A_{P'} \#*$
 $P' \rangle \langle A_{P'} \#* xvec \rangle$
 $\langle x \# (p \cdot xvec) \rangle \langle (p \cdot xvec) \#* xvec \rangle \langle (p \cdot xvec) \#* P' \rangle$
have $(x\#((p \cdot xvec) @ A_{P'})) \#* ((\nu x)(\nu * xvec)P')$
by $simp$
moreover from $\langle x \# X \rangle \langle (p \cdot xvec) \#* X \rangle \langle A_{P'} \#* X \rangle$
have $(x\#((p \cdot xvec) @ A_{P'})) \#* X$ **by** $simp$
moreover from $\langle x \# Y \rangle \langle (p \cdot xvec) \#* Y \rangle \langle A_{P'} \#* Y \rangle$
have $(x\#((p \cdot xvec) @ A_{P'})) \#* Y$ **by** $simp$

```

moreover from  $\langle x \# Z \rangle \langle (p \cdot xvec) \#* Z \rangle \langle A_{P'} \#* Z \rangle$ 
have  $(x\#((p \cdot xvec)@A_{P'})) \#* Z$  by simp
moreover from  $\langle x \# (p \cdot xvec) \rangle \langle x \# A_{P'} \rangle \langle (p \cdot xvec) \#* A_{P'} \rangle$ 
   $\langle distinct A_{P'} \rangle \langle distinct xvec \rangle \langle distinctPerm p \rangle$ 
have distinct  $(x\#((p \cdot xvec)@A_{P'}))$  by simp
ultimately show ?case
  by(rule cBrClose)
next
case(cScope  $\Psi P P' x A_P \Psi_P C X Y Z$ )
then obtain  $\Psi' A_{P'} \Psi_{P'}$  where FrP': extractFrame  $P' = \langle A_{P'}, \Psi_{P'} \rangle$  and
distinct  $A_{P'}$ 
  and  $\Psi_P \otimes \Psi' \simeq \Psi_{P'}$  and  $A_{P'} \#* C$  and  $A_{P'} \#* P'$ 
  and  $A_{P'} \#* (x\#X)$  and  $A_{P'} \#* Y$  and  $A_{P'} \#* Z$ 
using cScope(4)[where  $ba=x\#X$ ] by (metis freshStarAtom fresh-star-list-cons(1))
from  $\langle A_{P'} \#* (x\#X) \rangle$  have  $x \# A_{P'}$  and  $A_{P'} \#* X$  by simp+
moreover from FrP' have extractFrame( $(\nu x)P'$ ) =  $\langle (x\#A_{P'}), \Psi_{P'} \rangle$  by simp
moreover note  $\langle \Psi_P \otimes \Psi' \simeq \Psi_{P'} \rangle$ 
moreover from  $\langle x \# C \rangle \langle A_{P'} \#* C \rangle$  have  $(x\#A_{P'}) \#* C$  by simp
  moreover from  $\langle A_{P'} \#* P' \rangle$  have  $(x\#A_{P'}) \#* ((\nu x)P')$  by(simp add:
abs-fresh fresh-star-def)
moreover from  $\langle x \# X \rangle \langle A_{P'} \#* X \rangle$  have  $(x\#A_{P'}) \#* X$  by simp
moreover from  $\langle x \# Y \rangle \langle A_{P'} \#* Y \rangle$  have  $(x\#A_{P'}) \#* Y$  by simp
moreover from  $\langle x \# Z \rangle \langle A_{P'} \#* Z \rangle$  have  $(x\#A_{P'}) \#* Z$  by simp
moreover from  $\langle x \# A_{P'} \rangle \langle distinct A_{P'} \rangle$  have distinct( $x\#A_{P'}$ ) by simp
ultimately show ?case by(elim cScope)
next
case(cBang  $\Psi P P' A_P \Psi_P C B Y Z$ )
then obtain  $\Psi' A_{P'} \Psi_{P'}$  where FrP': extractFrame  $P' = \langle A_{P'}, \Psi_{P'} \rangle$ 
  and  $(\Psi_P \otimes \mathbf{1}) \otimes \Psi' \simeq \Psi_{P'}$ 
  and  $A_{P'} \#* C$  and  $A_{P'} \#* P'$  and distinct  $A_{P'}$ 
  and  $A_{P'} \#* B$  and  $A_{P'} \#* Y$  and  $A_{P'} \#* Z$ 
  using cBang by (metis psiFreshVec(4) psiFreshVec(7))
with  $\langle \Psi_P \simeq \mathbf{1} \rangle \langle (\Psi_P \otimes \mathbf{1}) \otimes \Psi' \simeq \Psi_{P'} \rangle$  have  $\mathbf{1} \otimes \Psi' \simeq \Psi_{P'}$ 
  by(metis Identity AssertionStatEqTrans composition' Commutativity Associativity AssertionStatEqSym)
then show ?case using FrP'  $\langle A_{P'} \#* P' \rangle \langle A_{P'} \#* C \rangle \langle A_{P'} \#* B \rangle \langle A_{P'} \#* Y \rangle$ 
 $\langle A_{P'} \#* Z \rangle \langle distinct A_{P'} \rangle$ 
  by(elim cBang)
qed
with  $A$  have ?thesis by simp
}
moreover have  $A_P \#* ([::name\ list])$  and  $A_P \#* ([::'b\ list])$  and  $A_P \#* ([::('a,$ 
'b, 'c) psi list]) by simp+
ultimately show ?thesis by blast
qed

lemma expandFrame:
fixes  $\Psi \quad :: 'b$ 
and  $P \quad :: ('a, 'b, 'c) psi$ 

```

and α :: 'a action
and P' :: ('a, 'b, 'c) psi
and A_P :: name list
and Ψ_P :: 'b
and C :: 'f::fs-name
and C' :: 'g::fs-name

assumes *Trans*: $\Psi \triangleright P \mapsto \alpha \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $bn \ \alpha \ \#* \ \text{subject } \alpha$
and *distinct*($bn \ \alpha$)
and $A_P \ \#* \ \alpha$
and $A_P \ \#* \ P$
and $A_P \ \#* \ C$
and $A_P \ \#* \ C'$
and $bn \ \alpha \ \#* \ P$
and $bn \ \alpha \ \#* \ C'$

obtains $p \ \Psi' \ A_{P'} \ \Psi_{P'}$ **where** $set \ p \subseteq set(bn \ \alpha) \times set(bn(p \cdot \alpha))$ **and** $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_{P'}$ **and** *distinctPerm* p **and**
extractFrame $P' = \langle A_{P'}, \Psi_{P'} \rangle$ **and** $A_{P'} \ \#* \ P'$ **and** $A_{P'} \ \#* \ \alpha$ **and** $A_{P'} \ \#* \ (p \cdot \alpha)$
and
 $A_{P'} \ \#* \ C$ **and** $(bn(p \cdot \alpha)) \ \#* \ C'$ **and** $(bn(p \cdot \alpha)) \ \#* \ \alpha$ **and** $(bn(p \cdot \alpha)) \ \#* \ P'$ **and**
distinct $A_{P'}$
using *assms*
apply(*cases* $\alpha = \tau$)
by(*auto intro: expandTauFrame[where C=C] expandNonTauFrame[where C=C*
and $C' = C'$)

abbreviation
frameImpJudge ($- \hookrightarrow_F - [80, 80] \ 80$)
where $F \hookrightarrow_F G \equiv \text{FrameStatImp } F \ G$

lemma *FrameStatEqImpCompose*:
fixes F :: 'b frame
and G :: 'b frame
and H :: 'b frame
and I :: 'b frame

assumes $F \simeq_F G$
and $G \hookrightarrow_F H$
and $H \simeq_F I$

shows $F \hookrightarrow_F I$
using *assms*
by(*auto simp add: FrameStatEq-def*) (*blast intro: FrameStatImpTrans*)

lemma *transferNonTauFrame*:


```

fixes  $\Psi_F$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\alpha$  :: 'a action
and  $P'$  :: ('a, 'b, 'c) psi
and  $A_F$  :: name list
and  $A_G$  :: name list
and  $\Psi_G$  :: 'b

assumes  $\Psi_F \triangleright P \mapsto \alpha \prec P'$ 
and extractFrame  $P = \langle A_P, \Psi_P \rangle$ 
and distinct  $A_P$ 
and distinct(bn  $\alpha$ )
and  $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$ 
and  $A_F \#* P$ 
and  $A_G \#* P$ 
and  $A_F \#*$  subject  $\alpha$ 
and  $A_G \#*$  subject  $\alpha$ 
and  $A_P \#* A_F$ 
and  $A_P \#* A_G$ 
and  $A_P \#* \Psi_F$ 
and  $A_P \#* \Psi_G$ 
and  $\alpha \neq \tau$ 

shows  $\Psi_G \triangleright P \mapsto \alpha \prec P'$ 
using assms
proof(nominal-induct  $\Psi_F P R s == \alpha \prec P' A_P \Psi_P$  avoiding:  $\alpha P' \Psi_G A_F A_G$  rule: semanticsFrameInduct)
case(cAlpha  $\Psi_F P A_P \Psi_P p \alpha P' \Psi_G A_F A_G$ )
from  $\langle \langle A_F, \Psi_F \otimes (p \cdot \Psi_P) \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes (p \cdot \Psi_P) \rangle \rangle$ 
have  $(p \cdot (\langle A_F, \Psi_F \otimes (p \cdot \Psi_P) \rangle)) \hookrightarrow_F (p \cdot (\langle A_G, \Psi_G \otimes (p \cdot \Psi_P) \rangle))$ 
by(rule FrameStatImpClosed)
with  $\langle A_P \#* A_F \rangle \langle (p \cdot A_P) \#* A_F \rangle \langle A_P \#* \Psi_F \rangle \langle (p \cdot A_P) \#* \Psi_F \rangle \langle A_P \#* A_G \rangle$ 
 $\langle (p \cdot A_P) \#* A_G \rangle \langle A_P \#* \Psi_G \rangle \langle (p \cdot A_P) \#* \Psi_G \rangle$ 
 $\langle \text{distinctPerm } p \rangle \langle \text{set } p \subseteq \text{set } A_P \times \text{set } (p \cdot A_P) \rangle$  have  $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F$ 
 $\langle A_G, \Psi_G \otimes \Psi_P \rangle$ 
by(simp add: eqvts)
with cAlpha show ?case by force
next
case(cInput  $\Psi_F M K \text{ xvec } N \text{ Tvec } P \alpha P' \Psi_G A_F A_G$ )
from cInput have  $A_F \#* K$  and  $A_G \#* K$  by(auto simp add: residualInject)

from  $\langle A_F \#* (M(\lambda * \text{xvec } N).P) \rangle \langle A_G \#* (M(\lambda * \text{xvec } N).P) \rangle$  have  $A_F \#* M$  and
 $A_G \#* M$  by simp+
from  $\langle \Psi_F \vdash M \leftrightarrow K \rangle$ 
have  $\Psi_F \otimes \mathbf{1} \vdash M \leftrightarrow K$ 
by(blast intro: statEqEnt Identity AssertionStatEqSym)
with  $\langle A_F \#* M \rangle \langle A_F \#* K \rangle$ 
have  $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow K$ 
by(force intro: frameImpI)

```

with $\langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle$
have $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow K$
by(*simp add: FrameStatEq-def FrameStatImp-def*)
with $\langle A_G \#* M \rangle \langle A_G \#* K \rangle$
have $\Psi_G \otimes \mathbf{1} \vdash M \leftrightarrow K$ **by**(*force dest: frameImpE*)
then have $\Psi_G \vdash M \leftrightarrow K$ **by**(*blast intro: statEqEnt Identity*)
then show *?case using* $\langle \text{distinct } xvec \rangle \langle \text{set } xvec \subseteq \text{supp } N \rangle \langle \text{length } xvec = \text{length } Tvec \rangle$ **using** *cInput Input*
by(*force simp add: residualInject*)
next
case(*cBrInput* $\Psi_F M K xvec N Tvec P \alpha P' \Psi_G A_F A_G$)
from *cBrInput* **have** $A_F \#* K$ **and** $A_G \#* K$ **by**(*auto simp add: residualInject*)

from $\langle A_F \#* (M(\lambda*xvec N).P) \rangle \langle A_G \#* (M(\lambda*xvec N).P) \rangle$ **have** $A_F \#* M$ **and** $A_G \#* M$ **by** *simp+*
from $\langle \Psi_F \vdash K \succeq M \rangle$
have $\Psi_F \otimes \mathbf{1} \vdash K \succeq M$
by(*blast intro: statEqEnt Identity AssertionStatEqSym*)
with $\langle A_F \#* M \rangle \langle A_F \#* K \rangle$
have $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F K \succeq M$
by(*force intro: frameImpI*)
with $\langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle$
have $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F K \succeq M$
by(*simp add: FrameStatEq-def FrameStatImp-def*)
with $\langle A_G \#* M \rangle \langle A_G \#* K \rangle$
have $\Psi_G \otimes \mathbf{1} \vdash K \succeq M$ **by**(*force dest: frameImpE*)
then have $\Psi_G \vdash K \succeq M$ **by**(*blast intro: statEqEnt Identity*)
then show *?case using* $\langle \text{distinct } xvec \rangle \langle \text{set } xvec \subseteq \text{supp } N \rangle \langle \text{length } xvec = \text{length } Tvec \rangle$ **using** *cBrInput BrInput*
by(*force simp add: residualInject*)
next
case(*cOutput* $\Psi_F M K N P \alpha P' \Psi_G A_F A_G$)
from *cOutput* **have** $A_F \#* K$ **and** $A_G \#* K$ **by**(*auto simp add: residualInject*)

from $\langle A_F \#* (M\langle N \rangle.P) \rangle \langle A_G \#* (M\langle N \rangle.P) \rangle$ **have** $A_F \#* M$ **and** $A_G \#* M$ **by** *simp+*
from $\langle \Psi_F \vdash M \leftrightarrow K \rangle$
have $\Psi_F \otimes \mathbf{1} \vdash M \leftrightarrow K$
by(*blast intro: statEqEnt Identity AssertionStatEqSym*)
with $\langle A_F \#* M \rangle \langle A_F \#* K \rangle$
have $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow K$
by(*force intro: frameImpI*)
with $\langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle$
have $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow K$
by(*simp add: FrameStatImp-def*)
with $\langle A_G \#* M \rangle \langle A_G \#* K \rangle$
have $\Psi_G \otimes \mathbf{1} \vdash M \leftrightarrow K$ **by**(*force dest: frameImpE*)
then have $\Psi_G \vdash M \leftrightarrow K$ **by**(*blast intro: statEqEnt Identity*)
then show *?case using* *cOutput Output* **by**(*force simp add: residualInject*)

```

next
  case(cBrOutput  $\Psi_F M K N P \alpha P' \Psi_G A_F A_G$ )
  from cBrOutput have  $A_F \#* K$  and  $A_G \#* K$  by(auto simp add: residualInject)

  from  $\langle A_F \#* (M \langle N \rangle . P) \rangle \langle A_G \#* (M \langle N \rangle . P) \rangle$  have  $A_F \#* M$  and  $A_G \#* M$  by
simp+
  from  $\langle \Psi_F \vdash M \preceq K \rangle$ 
  have  $\Psi_F \otimes \mathbf{1} \vdash M \preceq K$ 
  by(blast intro: statEqEnt Identity AssertionStatEqSym)
  with  $\langle A_F \#* M \rangle \langle A_F \#* K \rangle$ 
  have  $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F M \preceq K$ 
  by(force intro: frameImpI)
  with  $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$ 
  have  $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F M \preceq K$ 
  by(simp add: FrameStatImp-def)
  with  $\langle A_G \#* M \rangle \langle A_G \#* K \rangle$ 
  have  $\Psi_G \otimes \mathbf{1} \vdash M \preceq K$  by(force dest: frameImpE)
  then have  $\Psi_G \vdash M \preceq K$  by(blast intro: statEqEnt Identity)
  then show ?case using cBrOutput BrOutput by(force simp add: residualInject)
next
  case(cCase  $\Psi_F P \varphi Cs A_P \Psi_P \alpha P' \Psi_G A_F A_G$ )
  from  $\langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\langle A_F, \Psi_F \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1} \rangle$ 
  by(metis frameIntCompositionSym Identity AssertionStatEqTrans)
  moreover note  $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$ 
  moreover from  $\langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\langle A_G, \Psi_G \otimes \mathbf{1} \rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$ 
  by(metis frameIntCompositionSym Identity AssertionStatEqTrans Assertion-
StatEqSym)
  ultimately have  $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$ 
  by(rule FrameStatEqImpCompose)
  with cCase have  $\Psi_G \triangleright P \mapsto \alpha \prec P'$ 
  by(metis freshStarPair(2) memFreshChain(1) psiCasesFreshChain(1) psiFreshVec(3))
  moreover note  $\langle (\varphi, P) \in \text{set } Cs \rangle$ 
  moreover from  $\langle A_F \#* (Cases Cs) \rangle \langle A_G \#* (Cases Cs) \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$  have
 $A_F \#* \varphi$  and  $A_G \#* \varphi$ 
  by(auto dest: memFreshChain)
  from  $\langle \Psi_F \vdash \varphi \rangle$  have  $\Psi_F \otimes \mathbf{1} \vdash \varphi$  by(blast intro: statEqEnt Identity Assertion-
StatEqSym)
  with  $\langle A_F \#* \varphi \rangle$  have  $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F \varphi$  by(force intro: frameImpI)
  with  $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$  have  $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F \varphi$ 
  by(simp add: FrameStatImp-def)
  with  $\langle A_G \#* \varphi \rangle$  have  $\Psi_G \otimes \mathbf{1} \vdash \varphi$  by(force dest: frameImpE)
  then have  $\Psi_G \vdash \varphi$  by(blast intro: statEqEnt Identity)
  ultimately show ?case using  $\langle \text{guarded } P \rangle$  cCase Case by(force intro: residual-
Inject)
next
  case(cPar1  $\Psi_F \Psi_Q P \alpha P' A_Q Q A_P \Psi_P \alpha' PQ' \Psi_G A_F A_G$ )
  from  $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$  have  $A_F \#* P$  and  $A_G \#* P$  and  $A_F$ 
 $\#* Q$  and  $A_G \#* Q$ 
  by simp+

```

have FrQ : $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by fact**
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans
Commutativity frameResChainPres frameNilStatEq)
moreover note $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans
Commutativity frameResChainPres frameNilStatEq)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(rule FrameStatEqImpCompose)
moreover from $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle FrQ \langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle$ **have** A_F
 $\#* \Psi_Q$ **and** $A_G \#* \Psi_Q$
by(force dest: extractFrameFreshChain)+
moreover note $\langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_F \#* subject\ \alpha' \rangle \langle A_G \#* subject\ \alpha' \rangle \langle A_P$
 $\#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_G \rangle$
moreover from $\langle \alpha \prec P' \parallel Q = \alpha' \prec PQ' \rangle \langle bn\ \alpha \#* \alpha' \rangle$
obtain $p\ P''$ **where** $\alpha \prec P' = \alpha' \prec P''$ **and** $set\ p \subseteq set(bn\ \alpha') \times set(bn\ \alpha)$
and $PQ' = P'' \parallel (p \cdot Q)$
apply(drule-tac sym)
by(rule actionPar1Dest) (assumption | simp | blast dest: sym)+
ultimately have $\Psi_G \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'$ **using** $\langle \alpha' \neq \tau \rangle$ **by**(force intro:
cPar1)
then show ?case **using** $FrQ \langle (bn\ \alpha) \#* Q \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* P \rangle \langle A_Q \#* \alpha \rangle$
using cPar1
by(metis Par1)
next
case(cPar2 $\Psi_F \Psi_P Q \alpha Q' A_P P A_Q \Psi_Q \alpha' PQ' \Psi_G A_F A_G$)
from $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$ **have** $A_F \#* P$ **and** $A_G \#* P$ **and** A_F
 $\#* Q$ **and** $A_G \#* Q$
by simp+
have FrP : $extractFrame\ P = \langle A_P, \Psi_P \rangle$ **by fact**
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(metis Associativity frameResChainPres frameNilStatEq)
moreover note $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by(metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by(rule FrameStatEqImpCompose)
moreover from $\langle A_F \#* P \rangle \langle A_G \#* P \rangle FrP \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle$ **have** A_F
 $\#* \Psi_P$ **and** $A_G \#* \Psi_P$
by(force dest: extractFrameFreshChain)+
moreover note $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_F \#* subject\ \alpha' \rangle \langle A_G \#* subject\ \alpha' \rangle \langle A_Q$
 $\#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_G \rangle$
moreover from $\langle \alpha \prec P \parallel Q' = \alpha' \prec PQ' \rangle \langle bn\ \alpha \#* \alpha' \rangle$
obtain $p\ Q''$ **where** $\alpha \prec Q' = \alpha' \prec Q''$ **and** $set\ p \subseteq set(bn\ \alpha') \times set(bn\ \alpha)$
and $PQ' = (p \cdot P) \parallel Q''$
apply(drule-tac sym)
by(rule actionPar2Dest) (assumption | simp | blast dest: sym)+
ultimately have $\Psi_G \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'$ **using** $\langle \alpha' \neq \tau \rangle$ **by**(force intro:

cPar2)
then show *?case using FrP* $\langle (bn \ \alpha) \ \#* \ P \rangle \langle A_P \ \#* \ \Psi_G \rangle \langle A_P \ \#* \ Q \rangle \langle A_P \ \#* \ \alpha \rangle$
using *cPar2*
by(*metis Par2*)
next
case *cComm1*
then show *?case by*(*simp add: residualInject*)
next
case *cComm2*
then show *?case by*(*simp add: residualInject*)
next
case(*cBrMerge* $\Psi_F \ \Psi_Q \ P \ M \ N \ P' \ A_P \ \Psi_P \ Q \ Q' \ A_Q \ \alpha \ PQ' \ \Psi_G \ A_F \ A_G$)
from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have *FimpP*: $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)

from $\langle \iota M(N) \prec P' \parallel Q' = \alpha \prec PQ' \rangle$ **have** $\alpha = \iota M(N)$ **and** $(P' \parallel Q') = PQ'$
by(*simp add: residualInject*)+

moreover note *FimpP* $\langle A_F \ \#* \ (P \parallel Q) \rangle \langle A_G \ \#* \ (P \parallel Q) \rangle \langle A_F \ \#* \ subject \ \alpha \rangle$
 $\langle A_G \ \#* \ subject \ \alpha \rangle$
 $\langle A_P \ \#* \ A_F \rangle \langle A_P \ \#* \ A_G \rangle \langle A_P \ \#* \ \Psi_F \rangle \langle A_P \ \#* \ \Psi_G \rangle \langle A_P \ \#* \ \Psi_Q \rangle$
ultimately have *TransP*: $\Psi_G \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'$
by(*intro cBrMerge(2)[where bd=A_G]*) *auto*

from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *FimpQ*: $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)

note $\langle \alpha = \iota M(N) \rangle \langle (P' \parallel Q') = PQ' \rangle$ *FimpQ* $\langle A_F \ \#* \ (P \parallel Q) \rangle \langle A_G \ \#* \ (P \parallel Q) \rangle \langle A_F \ \#* \ subject \ \alpha \rangle \langle A_G \ \#* \ subject \ \alpha \rangle$
 $\langle A_Q \ \#* \ A_F \rangle \langle A_Q \ \#* \ A_G \rangle \langle A_Q \ \#* \ \Psi_F \rangle \langle A_Q \ \#* \ \Psi_G \rangle \langle A_Q \ \#* \ \Psi_P \rangle$
then have *TransQ*: $\Psi_G \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'$
by(*intro cBrMerge(6)[where bd=A_G]*) *auto*

have $\Psi_G \triangleright P \parallel Q \mapsto \iota M(N) \prec P' \parallel Q'$ **using** *TransP TransQ cBrMerge*
by(*intro BrMerge*)
with $\langle \iota M(N) \prec P' \parallel Q' = \alpha \prec PQ' \rangle$

show *?case by simp*
next
case(*cBrComm1* $\Psi_F \Psi_Q P M N P' A_P \Psi_P Q \text{ xvec } Q' A_Q \alpha PQ' \Psi_G A_F A_G$)
from $\langle \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle \langle A_F \#* \text{ subject } \alpha \rangle \langle A_G \#* \text{ subject } \alpha \rangle$
have $A_F \#* M$ **and** $A_G \#* M$
by(*auto simp add: residualInject*)

from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have *FimpP*: $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)

note *FimpP* $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle \langle A_F \#* M \rangle \langle A_G \#* M \rangle$
 $\langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_G \rangle \langle A_P \#* \Psi_Q \rangle$
then have *TransP*: $\Psi_G \otimes \Psi_Q \triangleright P \mapsto \text{!}M \langle N \rangle \prec P'$
by (*intro cBrComm1(4)[where bc=A_F and bd=A_G]*) *auto*
from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have *FimpQ*: $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
 Ψ_Q
by (*metis FrameStatImpTrans*)
note $\langle \text{distinct xvec} \rangle \text{!}M \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle \langle A_F \#* M \rangle \langle A_G \#* M \rangle$
 $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* \Psi_P \rangle$
then have *TransQ*: $\Psi_G \otimes \Psi_P \triangleright Q \mapsto \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec Q'$
by (*simp add: cBrComm1.hyps(8) freshCompChain(1)*)

from *TransP TransQ cBrComm1*
have $\Psi_G \triangleright P \parallel Q \mapsto \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q'$
by(*intro BrComm1*)
with $\langle \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle$
show *?case*
by *simp*
next
case(*cBrComm2* $\Psi_F \Psi_Q P M \text{ xvec } N P' A_P \Psi_P Q Q' A_Q \alpha PQ' \Psi_G A_F A_G$)
from $\langle \text{!}M(\nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle \langle A_F \#* \text{ subject } \alpha \rangle \langle A_G \#* \text{ subject } \alpha \rangle$
 α
have $A_F \#* M$ **and** $A_G \#* M$
by(*auto simp add: residualInject*)

from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have $FimpQ: \langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)

note $FimpQ \langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle \langle A_F \#* M \rangle \langle A_G \#* M \rangle$
 $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* \Psi_P \rangle$
then have $TransQ: \Psi_G \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\downarrow N) \prec Q'$
by(*intro cBrComm2(8)[where bc=A_F and bd=A_G]*) *auto*

from $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
ultimately have $FimpP: \langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)

note $\langle distinct\ xvec \rangle FimpP \langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle \langle A_F \#* M \rangle \langle A_G \#* M \rangle$
 $\langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_G \rangle \langle A_P \#* \Psi_Q \rangle$
then have $TransP: \Psi_G \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P'$
by(*intro cBrComm2(4)[where bc=A_F and bd=A_G]*) *auto*

from $TransP\ TransQ\ cBrComm2$
have $\Psi_G \triangleright P \parallel Q \mapsto \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P' \parallel Q'$
by(*intro BrComm2*)
with $\langle \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P' \parallel Q' = \alpha \prec PQ' \rangle$
show *?case*
by *simp*

next

case *cBrClose*

then show *?case by(simp add: residualInject)*

next

case(*cOpen* $\Psi_F\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ \alpha\ P''\ \Psi_G\ A_F\ A_G$)

from $\langle M(\downarrow \nu * (xvec @ x \# yvec)) \langle N \rangle \prec P' = \alpha \prec P'' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# \alpha \rangle \langle x \# P'' \rangle \langle distinct(bn\ \alpha) \rangle$

obtain $xvec'\ x'\ yvec'\ N'$ **where** $M(\downarrow \nu * (xvec @ yvec)) \langle N \rangle \prec P' = M(\downarrow \nu * (xvec' @ yvec')) \langle N' \rangle \langle [(x, x')] \cdot N' \rangle \prec [(x, x')] \cdot P''$

and $\alpha = M(\downarrow \nu * (xvec' @ x' \# yvec')) \langle N' \rangle$

apply(*cases rule: actionCases[where $\alpha = \alpha$]*)

apply(*simp add: residualInject*)

apply(*simp add: residualInject*)

```

    apply(simp add: residualInject)
    apply(metis boundOutputOpenDest)
    apply(simp add: residualInject)
    by(simp add: residualInject)

  then have  $\Psi_G \triangleright P \mapsto M(\nu^*(xvec@yvec))\langle N \rangle \prec P'$  using cOpen
    by(intro cOpen(4)[where  $bc=A_F$  and  $bd=A_G$ ]) auto
  with  $\langle x \in \text{supp } N \rangle \langle x \# \Psi_G \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle$ 
  have  $\Psi_G \triangleright (\nu x)P \mapsto M(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P'$ 
    by(intro Open)
  then show ?case using  $\langle \alpha = M(\nu^*(xvec'@x'\#yvec'))\langle N' \rangle \rangle \langle M(\nu^*(xvec @ x \# yvec))\langle N \rangle \prec P' = \alpha \prec P'' \rangle$ 
    by simp
next
  case(cBrOpen  $\Psi_F P M xvec yvec N P' x A_P \Psi_P \alpha P'' \Psi_G A_F A_G$ )
  from  $\langle jM(\nu^*(xvec @ x \# yvec))\langle N \rangle \prec P' = \alpha \prec P'' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# \alpha \rangle \langle x \# P'' \rangle \langle \text{distinct}(bn \alpha) \rangle$ 
  obtain  $xvec' x' yvec' N'$  where  $jM(\nu^*(xvec@yvec))\langle N \rangle \prec P' = jM(\nu^*(xvec'@yvec'))\langle ([x, x'] \cdot N') \rangle \prec ([x, x'] \cdot P'')$ 
    and  $\alpha = jM(\nu^*(xvec'@x'\#yvec'))\langle N' \rangle$ 
  apply(cases rule: actionCases[where  $\alpha=\alpha$ ])
  apply(simp add: residualInject)
  apply(simp add: residualInject)
  apply(simp add: residualInject)
  apply(simp add: residualInject)
  apply(metis boundOutputOpenDest)
  by(simp add: residualInject)

  then have  $\Psi_G \triangleright P \mapsto jM(\nu^*(xvec@yvec))\langle N \rangle \prec P'$  using cBrOpen
    by(intro cBrOpen) (assumption | simp)+
  with  $\langle x \in \text{supp } N \rangle \langle x \# \Psi_G \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle$ 
  have  $\Psi_G \triangleright (\nu x)P \mapsto jM(\nu^*(xvec@x\#yvec))\langle N \rangle \prec P'$ 
    by(intro BrOpen)
  then show ?case using  $\langle \alpha = jM(\nu^*(xvec'@x'\#yvec'))\langle N' \rangle \rangle \langle jM(\nu^*(xvec @ x \# yvec))\langle N \rangle \prec P' = \alpha \prec P'' \rangle$ 
    by simp
next
  case(cScope  $\Psi_F P \alpha P' x A_P \Psi_P \alpha' xP \Psi_G A_F A_G$ )
  from  $\langle \alpha \prec (\nu x)P' = \alpha' \prec xP \rangle \langle x \# \alpha \rangle \langle x \# \alpha' \rangle$  obtain  $P''$  where  $xP = (\nu x)P''$ 
  and  $\alpha \prec P' = \alpha' \prec P''$ 
  by(drule-tac sym) (force intro: actionScopeDest)
  then have  $\Psi_G \triangleright P \mapsto \alpha \prec P'$  using cScope by auto
  with  $\langle x \# \Psi_G \rangle \langle x \# \alpha' \rangle \langle \alpha \prec P' = \alpha' \prec P'' \rangle \langle xP = (\nu x)P'' \rangle$  show ?case
    by(metis Scope)
next
  case(cBang  $\Psi_F P A_P \Psi_P \alpha P' \Psi_G A_F A_G$ )
  from  $\langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1} \rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1} \rangle$ 
  by(metis frameIntCompositionSym Identity AssertionStatEqTrans)
  moreover note  $\langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle$ 

```


moreover from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\langle A_G, \Psi_G \otimes \mathbf{1} \rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1} \rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1} \rangle$
by(*rule FrameStatEqImpCompose*)
with *cBang* **have** $\Psi_G \triangleright P \parallel !P \mapsto \alpha \prec P'$ **by force**
then show *?case* **using** $\langle \text{guarded } P \rangle$ **using** *cBang* **by**(*metis Bang*)
qed

lemma *transferTauFrame*:

fixes $\Psi_F :: 'b$
and $P :: ('a, 'b, 'c)$ *psi*
and $P' :: ('a, 'b, 'c)$ *psi*
and $A_F :: \text{name list}$
and $A_G :: \text{name list}$
and $\Psi_G :: 'b$

assumes $\Psi_F \triangleright P \mapsto_{\tau} \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
and $A_F \#^* P$
and $A_G \#^* P$
and $A_P \#^* A_F$
and $A_P \#^* A_G$
and $A_P \#^* \Psi_F$
and $A_P \#^* \Psi_G$

shows $\Psi_G \triangleright P \mapsto_{\tau} \prec P'$

using *assms*

proof(*nominal-induct avoiding: \Psi_G A_F A_G rule: tauFrameInduct*)

case(*cAlpha \Psi_F P P' A_P \Psi_P p \Psi_G A_F A_G*)

from $\langle \langle A_F, \Psi_F \otimes (p \cdot \Psi_P) \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes (p \cdot \Psi_P) \rangle \rangle$

have $\langle p \cdot (\langle A_F, \Psi_F \otimes (p \cdot \Psi_P) \rangle) \rangle \hookrightarrow_F \langle p \cdot (\langle A_G, \Psi_G \otimes (p \cdot \Psi_P) \rangle) \rangle$

by(*rule FrameStatImpClosed*)

with $\langle A_P \#^* A_F \rangle \langle (p \cdot A_P) \#^* A_F \rangle \langle A_P \#^* \Psi_F \rangle \langle (p \cdot A_P) \#^* \Psi_F \rangle \langle A_P \#^* A_G \rangle$
 $\langle (p \cdot A_P) \#^* A_G \rangle \langle A_P \#^* \Psi_G \rangle \langle (p \cdot A_P) \#^* \Psi_G \rangle$

$\langle \text{distinctPerm } p \rangle \langle \text{set } p \subseteq \text{set } A_P \times \text{set } (p \cdot A_P) \rangle$ **have** $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$

by(*simp add: eqvts*)

with *cAlpha* **show** *?case* **by** *blast*

next

case(*cCase \Psi_F P P' \varphi Cs A_P \Psi_P \Psi_G A_F A_G*)

from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\langle A_F, \Psi_F \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1} \rangle$

by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)

moreover note $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$

moreover from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\langle A_G, \Psi_G \otimes \mathbf{1} \rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$

by(*metis frameIntCompositionSym Identity AssertionStatEqTrans AssertionStatEqSym*)

ultimately have $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
by(rule *FrameStatEqImpCompose*)
with $cCase$ **have** $\Psi_G \triangleright P \mapsto_{\tau} \prec P'$
by (metis *freshStarPair*(2) *memFreshChain*(1) *psiCasesFreshChain*(1) *psiFreshVec*(3))
moreover note $\langle \varphi, P \rangle \in set\ Cs$
moreover from $\langle A_F \#* (Cases\ Cs) \rangle \langle A_G \#* (Cases\ Cs) \rangle \langle \varphi, P \rangle \in set\ Cs$ **have**
 $A_F \#* \varphi$ **and** $A_G \#* \varphi$
by(auto *dest: memFreshChain*)
from $\langle \Psi_F \vdash \varphi \rangle$ **have** $\Psi_F \otimes \mathbf{1} \vdash \varphi$ **by**(blast *intro: statEqEnt Identity Assertion-StatEqSym*)
with $\langle A_F \#* \varphi \rangle$ **have** $(\langle A_F, \Psi_F \otimes \mathbf{1} \rangle) \vdash_F \varphi$ **by**(force *intro: frameImpI*)
with $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$ **have** $(\langle A_G, \Psi_G \otimes \mathbf{1} \rangle) \vdash_F \varphi$
by(simp *add: FrameStatImp-def*)
with $\langle A_G \#* \varphi \rangle$ **have** $\Psi_G \otimes \mathbf{1} \vdash \varphi$ **by**(force *dest: frameImpE*)
then have $\Psi_G \vdash \varphi$ **by**(blast *intro: statEqEnt Identity*)
ultimately show *?case* **using** $\langle guarded\ P \rangle$ **by**(rule *Case*)
next
case(*cPar1* $\Psi_F \Psi_Q P P' A_Q Q A_P \Psi_P \Psi_G A_F A_G$)
from $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$ **have** $A_F \#* P$ **and** $A_G \#* P$ **and** A_F
 $\#* Q$ **and** $A_G \#* Q$
by *simp+*
have *IH*: $\bigwedge \Psi A_F A_G. \llbracket \langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi \otimes \Psi_P \rangle; A_F \#* P; A_G \#* P; A_P \#* A_F; A_P \#* A_G; A_P \#* (\Psi_F \otimes \Psi_Q); A_P \#* \Psi \rrbracket \implies \Psi$
 $\triangleright P \mapsto_{\tau} \prec P'$
by *fact*
have *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(metis *Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover note $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(metis *Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(rule *FrameStatEqImpCompose*)
moreover from $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle$ *FrQ* $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle$ **have** A_F
 $\#* \Psi_Q$ **and** $A_G \#* \Psi_Q$
by(force *dest: extractFrameFreshChain*)+
moreover note $\langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P$
 $\#* \Psi_Q \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_G \rangle$
ultimately have $\Psi_G \otimes \Psi_Q \triangleright P \mapsto_{\tau} \prec P'$
using *IH* **by** *blast*
then show *?case* **using** *FrQ* $\langle A_Q \#* \Psi_G \rangle \langle A_Q \#* P \rangle$
by(*intro Par1*) *auto*
next
case(*cPar2* $\Psi_F \Psi_P Q Q' A_P P A_Q \Psi_Q \Psi_G A_F A_G$)
from $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$ **have** $A_F \#* P$ **and** $A_G \#* P$ **and** A_F
 $\#* Q$ **and** $A_G \#* Q$

by *simp+*
 have *IH*: $\bigwedge \Psi A_F A_G. \llbracket \langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi \otimes \Psi_Q \rangle; A_F \#* Q; A_G \#* Q; A_Q \#* A_F; A_Q \#* A_G; A_Q \#* (\Psi_F \otimes \Psi_P); A_Q \#* \Psi \rrbracket \implies \Psi$
 $\triangleright Q \mapsto_{\tau} \prec Q'$
 by *fact*
 have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ by *fact*
 have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
 by(*metis Associativity frameResChainPres frameNilStatEq*)
 moreover note $\langle \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
 moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
 ultimately have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by(*rule FrameStatEqImpCompose*)
 moreover from $\langle A_F \#* P \rangle \langle A_G \#* P \rangle$ *FrP* $\langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle$ have $A_F \#* \Psi_P$ and $A_G \#* \Psi_P$
 by(*force dest: extractFrameFreshChain*)+
 moreover note $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_G \rangle$
 ultimately have $\Psi_G \otimes \Psi_P \triangleright Q \mapsto_{\tau} \prec Q'$
 using *IH* by *blast*
 then show *?case* using *FrP* $\langle A_P \#* \Psi_G \rangle \langle A_P \#* Q \rangle$
 by(*intro Par2*) *auto*
 next
 case(*cComm1* $\Psi_F \Psi_Q P M N P' A_P \Psi_P Q K \text{vec} Q' A_Q \Psi_G A_F A_G$)
 have *FimpG*: $\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$ by *fact*
 from $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$ have $A_F \#* P$ and $A_G \#* P$ and $A_F \#* Q$ and $A_G \#* Q$
 by *simp+*
 have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ by *fact*
 have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ by *fact*
 from $\langle A_F \#* P \rangle \langle A_G \#* P \rangle$ *FrP* $\langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle$ have $A_F \#* \Psi_P$ and $A_G \#* \Psi_P$
 by(*force dest: extractFrameFreshChain*)+
 from $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle$ *FrQ* $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle$ have $A_F \#* \Psi_Q$ and $A_G \#* \Psi_Q$
 by(*force dest: extractFrameFreshChain*)+
 from $\langle \Psi_F \otimes \Psi_P \triangleright Q \mapsto_{K'} (\nu * \text{vec}) \langle N \rangle \prec Q' \rangle$ have $\Psi_F \otimes \Psi_P \triangleright Q \mapsto_{R \text{Out}} K ((\nu * \text{vec}) N \prec' Q')$
 by(*simp add: residualInject*)
 with *FrQ* $\langle \text{distinct } A_Q \rangle$
 obtain K' where *KeqK'*: $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ and $A_P \#* K'$ and $A_F \#* K'$ and $A_G \#* K'$
 using $\langle A_P \#* Q \rangle \langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_P \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle \text{vec} \#* K \rangle \langle \text{distinct } \text{vec} \rangle$
 by(*elim outputObtainPrefix*[*where* $B = A_P \otimes A_F \otimes A_G$]) *force+*
 have $\Psi_G \otimes \Psi_Q \triangleright P \mapsto_{K'} (\nu * \text{vec}) \langle N \rangle \prec P'$
 proof –

from $\text{Keq}K'$ **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash K \leftrightarrow K'$ **by**(*rule statEqEnt[OF Associativity]*)
with $\langle \Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K \rangle$ **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then have $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with $\langle \Psi_F \otimes \Psi_Q \triangleright P \mapsto M(\downarrow N) \prec P' \rangle$ $\text{Fr}P$ $\langle \text{distinct } A_P \rangle$
have $\Psi_F \otimes \Psi_Q \triangleright P \mapsto K'(\downarrow N) \prec P'$ **using** $\langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K' \rangle$
by(*force intro: inputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
proof –
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEqTrans Commutativity frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed
ultimately show *?thesis* **using** $\langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_F \#* K' \rangle$
 $\langle A_G \#* K' \rangle \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_G \rangle \langle A_P \#* \Psi_Q \rangle$
 $\text{Fr}P$ $\langle \text{distinct } A_P \rangle$
by(*auto intro: transferNonTauFrame*)
qed

moreover from $\text{Fr}P$ $\langle \Psi_F \otimes \Psi_Q \triangleright P \mapsto M(\downarrow N) \prec P' \rangle$ $\langle \text{distinct } A_P \rangle$
obtain M' **where** $\text{Meq}M'$: $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $A_Q \#* M'$ **and**
 $A_F \#* M'$ **and** $A_G \#* M'$
using $\langle A_Q \#* P \rangle \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_P \#* \Psi_F \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle$
by(*elim inputObtainPrefix[where B=A_Q@A_F@A_G]*) *force+*

have $\Psi_G \otimes \Psi_P \triangleright Q \mapsto M'(\downarrow \nu * \text{xvec}) \langle N \rangle \prec Q'$
proof –
from $\text{Meq}M'$ **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash M \leftrightarrow M'$
by(*rule statEqEnt[OF Associativity]*)
with $\langle \Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K \rangle$ **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash K \leftrightarrow M'$
by(*blast intro: chanEqTrans chanEqSym compositionSym Commutativity statEqEnt*)
then have $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro: statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with $\langle \Psi_F \otimes \Psi_P \triangleright Q \mapsto K(\downarrow \nu * \text{xvec}) \langle N \rangle \prec Q' \rangle$ $\text{Fr}Q$ $\langle \text{distinct } A_Q \rangle$
have $\Psi_F \otimes \Psi_P \triangleright Q \mapsto M'(\downarrow \nu * \text{xvec}) \langle N \rangle \prec Q'$ **using** $\langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_P \rangle$
 $\langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$
by(*force intro: outputRenameSubject*)

moreover have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
proof –
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately show *?thesis using FimpG*
by(*elim FrameStatEqImpCompose*)
qed

ultimately show *?thesis using* $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_F \#* M' \rangle \langle A_G \#* M' \rangle$
 $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* \Psi_P \rangle \text{FrQ} \langle \text{distinct } A_Q \rangle \langle \text{distinct } xvec \rangle$
by(*auto intro: transferNonTauFrame*)
qed

moreover have $\Psi_G \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
proof –
from *MeqM'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M' \leftrightarrow M$
by(*blast intro: chanEqSym Associativity statEqEnt Commutativity compositionSym*)
moreover from *KeqK'* **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow K'$
by(*blast intro: chanEqSym Associativity statEqEnt Commutativity compositionSym*)
ultimately have $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$ **using** $\langle \Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$
by(*blast intro: chanEqSym chanEqTrans*)
then show *?thesis using* $\langle A_F \#* M' \rangle \langle A_F \#* K' \rangle \langle A_G \#* M' \rangle \langle A_G \#* K' \rangle$
FimpG
apply(*simp add: FrameStatImp-def*)
apply(*erule allE[where x=SChanEq' K' M']*)
by(*force intro: frameImpI dest: frameImpE*)
qed

ultimately show *?case using* $\langle A_P \#* \Psi_G \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_Q \rangle$
 $\langle A_P \#* K' \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M' \rangle \langle xvec \#* P \rangle \text{FrP FrQ}$
by(*intro Comm1*) (*assumption | simp*)+
next
case(*cComm2* $\Psi_F \Psi_Q P M xvec N P' A_P \Psi_P Q K Q' A_Q \Psi_G A_F A_G$)
have *FimpG*: $\langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle$ **by** *fact*
from $\langle A_F \#* (P \parallel Q) \rangle \langle A_G \#* (P \parallel Q) \rangle$ **have** $A_F \#* P$ **and** $A_G \#* P$ **and** $A_F \#* Q$ **and** $A_G \#* Q$
by *simp+*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_F \#* P \rangle \langle A_G \#* P \rangle \text{FrP} \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle$ **have** $A_F \#* \Psi_P$ **and** $A_G \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)+

from $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \text{Fr}Q \langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle$ **have** $A_F \#* \Psi_Q$ **and**
 $A_G \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)
from $\langle \Psi_F \otimes \Psi_P \triangleright Q \mapsto K \langle N \rangle \prec Q' \rangle \text{Fr}Q \langle \text{distinct } A_Q \rangle$
obtain K' **where** $\text{Keq}K'$: $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow K'$ **and** $A_P \#* K'$ **and** A_F
 $\#* K'$ **and** $A_G \#* K'$
using $\langle A_P \#* Q \rangle \langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_Q \#* \Psi_F \rangle$
 $\langle A_Q \#* \Psi_P \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle$
by(*elim inputObtainPrefix[where B=A_P@A_F@A_G]*) **force+**
have $\Psi_G \otimes \Psi_Q \triangleright P \mapsto K' \langle \nu * xvec \rangle \langle N \rangle \prec P'$
proof –
from $\text{Keq}K'$ **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash K \leftrightarrow K'$
by(*rule statEqEnt[OF Associativity]*)
with $\langle \Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K \rangle$ **have** $\Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K'$
by(*rule chanEqTrans*)
then have $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by(*metis statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans*
compositionSym Commutativity)
with $\langle \Psi_F \otimes \Psi_Q \triangleright P \mapsto M \langle \nu * xvec \rangle \langle N \rangle \prec P' \rangle \text{Fr}P \langle \text{distinct } A_P \rangle$
have $\Psi_F \otimes \Psi_Q \triangleright P \mapsto K' \langle \nu * xvec \rangle \langle N \rangle \prec P'$ **using** $\langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K' \rangle$
by(*force intro: outputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
proof –
have $\langle A_F, (\Psi_F \otimes \Psi_Q) \otimes \Psi_P \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEq-*
Trans Commutativity frameResChainPres frameNilStatEq)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_Q) \otimes \Psi_P \rangle$
by(*metis Associativity Composition AssertionStatEqSym AssertionStatEq-*
Trans Commutativity frameResChainPres frameNilStatEq)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed
ultimately show *?thesis* **using** $\langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_F \#* K' \rangle$
 $\langle A_G \#* K' \rangle \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_G \rangle \langle A_P \#* \Psi_Q \rangle$
 $\text{Fr}P \langle \text{distinct } A_P \rangle$
 $\langle \text{distinct } xvec \rangle$
by(*auto intro: transferNonTauFrame*)
qed

moreover from $\langle \Psi_F \otimes \Psi_Q \triangleright P \mapsto M \langle \nu * xvec \rangle \langle N \rangle \prec P' \rangle$ **have** $\Psi_F \otimes \Psi_Q \triangleright$
 $P \mapsto \text{R}Out M \langle \nu * xvec \rangle \langle N \rangle \prec P'$
by(*simp add: residualInject*)
moreover with $\text{Fr}P \langle \text{distinct } A_P \rangle$
obtain M' **where** $\text{Meq}M'$: $(\Psi_F \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $A_Q \#* M'$ **and**
 $A_F \#* M'$ **and** $A_G \#* M'$
using $\langle A_Q \#* P \rangle \langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_F \#* P \rangle \langle A_G \#* P \rangle \langle A_P \#* \Psi_F \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle xvec \#* M \rangle \langle \text{distinct } xvec \rangle$
by(*elim outputObtainPrefix[where B=A_Q@A_F@A_G]*) **force+**

have $\Psi_G \otimes \Psi_P \triangleright Q \mapsto M'(N) \prec Q'$
proof –
from $MeqM'$ **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash M \leftrightarrow M'$ **by**(*rule statEqEnt[OF Associativity]*)
with $\langle \Psi_F \otimes (\Psi_P \otimes \Psi_Q) \vdash M \leftrightarrow K \rangle$ **have** $\Psi_F \otimes (\Psi_Q \otimes \Psi_P) \vdash K \leftrightarrow M'$
by(*blast intro: chanEqTrans chanEqSym compositionSym Commutativity statEqEnt*)
then have $(\Psi_F \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro: statEqEnt AssertionStatEqSym Associativity AssertionStatEqTrans compositionSym Commutativity*)
with $\langle \Psi_F \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q' \rangle$ *FrQ* $\langle \text{distinct } A_Q \rangle$
have $\Psi_F \otimes \Psi_P \triangleright Q \mapsto M'(N) \prec Q'$ **using** $\langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$
by(*force intro: inputRenameSubject*)
moreover have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
proof –
have $\langle A_F, (\Psi_F \otimes \Psi_P) \otimes \Psi_Q \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi_P \otimes \Psi_Q \rangle$
by(*metis Associativity frameResChainPres frameNilStatEq*)
moreover have $\langle A_G, \Psi_G \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle A_G, (\Psi_G \otimes \Psi_P) \otimes \Psi_Q \rangle$
by(*metis Associativity AssertionStatEqSym frameResChainPres frameNilStatEq*)
ultimately show *?thesis* **using** *FimpG*
by(*elim FrameStatEqImpCompose*)
qed

ultimately show *?thesis* **using** $\langle A_F \#* Q \rangle \langle A_G \#* Q \rangle \langle A_F \#* M' \rangle \langle A_G \#* M' \rangle$
 $\langle A_Q \#* A_F \rangle \langle A_Q \#* A_G \rangle \langle A_Q \#* \Psi_F \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* \Psi_P \rangle$ *FrQ* $\langle \text{distinct } A_Q \rangle \langle \text{distinct } xvec \rangle$
by(*auto intro: transferNonTauFrame*)
qed

moreover have $\Psi_G \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$
proof –
from $MeqM'$ **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M' \leftrightarrow M$
by(*blast intro: chanEqSym Associativity statEqEnt Commutativity compositionSym*)
moreover from $KeqK'$ **have** $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow K'$
by(*blast intro: chanEqSym Associativity statEqEnt Commutativity compositionSym*)
ultimately have $\Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash K' \leftrightarrow M'$ **using** $\langle \Psi_F \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$
by(*blast intro: chanEqSym chanEqTrans*)
then show *?thesis* **using** $\langle A_F \#* M' \rangle \langle A_F \#* K' \rangle \langle A_G \#* M' \rangle \langle A_G \#* K' \rangle$
FimpG
apply(*simp add: FrameStatImp-def*)
apply(*erule alle[where x=SChanEq' K' M']*)
by(*force intro: frameImpI dest: frameImpE*)
qed

ultimately show $?case$ **using** $\langle A_P \#* \Psi_G \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_Q \rangle$
 $\langle A_P \#* K' \rangle \langle A_Q \#* \Psi_G \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M' \rangle \langle xvec \#* Q \rangle FrP FrQ$
by(*intro Comm2*) (*assumption* | *simp*)+
next
case(*cBrClose* $\Psi_F P M xvec N P' A_P \Psi_P x \Psi_G A_F A_G$)
from $\langle \Psi_F \triangleright P \mapsto jM(\nu*xvec)\langle N \rangle \prec P' \rangle$
have *suppM*: $((supp M)::name\ set) \subseteq ((supp P)::name\ set)$
by(*simp add: residualInject brOutputTermSupp*)

note $\langle \Psi_F \triangleright P \mapsto jM(\nu*xvec)\langle N \rangle \prec P' \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle distinct A_P \rangle \langle distinct xvec \rangle \langle \langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle \rangle$

moreover from $\langle x \# A_F \rangle \langle A_F \#* (\nu x)P \rangle \langle x \# A_G \rangle \langle A_G \#* (\nu x)P \rangle$
have $A_F \#* P$ **and** $A_G \#* P$ **by** *simp+*
moreover with *suppM*
have $A_F \#* M$ **and** $A_G \#* M$
by(*auto simp add: fresh-star-def fresh-def*)
moreover note $\langle A_P \#* A_F \rangle \langle A_P \#* A_G \rangle \langle A_P \#* \Psi_F \rangle \langle A_P \#* \Psi_G \rangle$
ultimately have $\Psi_G \triangleright P \mapsto jM(\nu*xvec)\langle N \rangle \prec P'$
by(*simp add: transferNonTauFrame*)
with $\langle x \in supp M \rangle \langle x \# \Psi_G \rangle$
show $?case$
by(*simp add: BrClose*)
next
case(*cScope* $\Psi_F P P' x A_P \Psi_P \Psi_G A_F A_G$)
then have $\Psi_G \triangleright P \mapsto \tau \prec P'$ **by** *auto*
with $\langle x \# \Psi_G \rangle$ **show** $?case$
by(*intro Scope*) *auto*
next
case(*cBang* $\Psi_F P P' A_P \Psi_P \Psi_G A_F A_G$)
from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1} \rangle \simeq_F \langle A_F, \Psi_F \otimes \mathbf{1} \rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans*)
moreover note $\langle \langle A_F, \Psi_F \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \mathbf{1} \rangle \rangle$
moreover from $\langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\langle A_G, \Psi_G \otimes \mathbf{1} \rangle \simeq_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1} \rangle$
by(*metis frameIntCompositionSym Identity AssertionStatEqTrans Assertion-StatEqSym*)
ultimately have $\langle A_F, \Psi_F \otimes \Psi_P \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \otimes \mathbf{1} \rangle$
by(*rule FrameStatEqImpCompose*)
with *cBang* **have** $\Psi_G \triangleright P \parallel !P \mapsto \tau \prec P'$ **by** *force*
then show $?case$ **using** $\langle guarded P \rangle$ **by**(*rule Bang*)
qed

lemma *transferFrame*:

fixes $\Psi_F :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c) psi$
and $A_F :: name\ list$


```

    and  $A_G$  :: name list
    and  $\Psi_G$  :: 'b

assumes  $\Psi_F \triangleright P \mapsto \alpha \prec P'$ 
    and  $\text{extractFrame } P = \langle A_P, \Psi_P \rangle$ 
    and  $\text{distinct } A_P$ 
    and  $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$ 
    and  $A_F \#^* P$ 
    and  $A_G \#^* P$ 
    and  $A_F \#^* \text{subject } \alpha$ 
    and  $A_G \#^* \text{subject } \alpha$ 
    and  $A_P \#^* A_F$ 
    and  $A_P \#^* A_G$ 
    and  $A_P \#^* \Psi_F$ 
    and  $A_P \#^* \Psi_G$ \Psi_G \triangleright P \mapsto \alpha \prec P'
    using assms
proof -
    from  $\langle \Psi_F \triangleright P \mapsto \alpha \prec P' \rangle$  have  $\text{distinct}(bn \ \alpha)$  by(auto dest: boundOutputDistinct)
    then show ?thesis using assms
        by(cases  $\alpha = \tau$ ) (auto intro: transferNonTauFrame transferTauFrame)
qed

lemma parCasesInputFrame[consumes 7, case-names cPar1 cPar2]:
    fixes  $\Psi$  :: 'b
    and  $P$  :: ('a, 'b, 'c) psi
    and  $Q$  :: ('a, 'b, 'c) psi
    and  $M$  :: 'a
    and xvec :: name list
    and  $N$  :: 'a
    and  $T$  :: ('a, 'b, 'c) psi
    and  $C$  :: 'd::fs-name

assumes Trans:  $\Psi \triangleright P \parallel Q \mapsto M(N) \prec T$ 
    and  $\text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle$ 
    and  $\text{distinct } A_{PQ}$ 
    and  $A_{PQ} \#^* \Psi$ 
    and  $A_{PQ} \#^* P$ 
    and  $A_{PQ} \#^* Q$ 
    and  $A_{PQ} \#^* M$ 
    and rPar1:  $\bigwedge P' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_P; \text{distinct } A_Q; A_P \#^* \Psi; A_P \#^* P; A_P \#^* Q; A_P \#^* M; A_Q \#^* \Psi; A_Q \#^* P; A_Q \#^* Q; A_Q \#^* M; A_P \#^* \Psi_Q; A_Q \#^* \Psi_P; A_P \#^* A_Q; A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \rrbracket \implies \text{Prop } (P' \parallel Q)$ 
    and rPar2:  $\bigwedge Q' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto M(N) \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{distinct } A_P; \text{distinct } A_Q; A_P \#^* \Psi; A_P \#^* P; A_P \#^* Q; A_P \#^* M; A_Q \#^* \Psi; A_Q \#^* P; A_Q \#^* Q; A_Q \#^* M; A_P \#^* \Psi_Q; A_Q \#^* \Psi_P; A_P \#^* A_Q; A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \rrbracket \implies \text{Prop } (P' \parallel Q)$ 

```

$P = \langle A_P, \Psi_P \rangle$; *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$;
 $\text{distinct } A_P$; $\text{distinct } A_Q$; $A_P \#* \Psi$; $A_P \#* P$; $A_P \#*$
 Q ; $A_P \#* M$; $A_Q \#* \Psi$; $A_Q \#* P$; $A_Q \#* Q$; $A_Q \#* M$;
 $A_P \#* \Psi_Q$; $A_Q \#* \Psi_P$; $A_P \#* A_Q$; $A_{PQ} = A_P @ A_Q$;
 $\Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies \text{Prop } (P \parallel Q')$
shows Prop T
using Trans
proof(*induct rule: parInputCases*[of - - - - - (A_{PQ}, Ψ_{PQ})])
case(*cPar1* $P' A_Q \Psi_Q$)
from $\langle A_Q \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_{PQ}$ **by simp+**
obtain $A_P \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
 $A_P \#* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \#* P$ **and** $A_P \#* Q$ **and** $A_P \#* \Psi$ **and** $A_P \#* M$ **and** $A_P \#* A_Q$
and $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_Q$
by simp+

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by fact**

from $\langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle$ *FrP* **have** $A_Q \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ *FrP* *FrQ* $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by simp**
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** *distinct*($A_P @ A_Q$)
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S : $\text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and *distinctPerm* p
and *Ψeq*: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq*: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption* | *simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\lfloor N \rfloor) \prec P' \rangle$ S $\langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_{PQ}$
 $\#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle$ *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto M(\lfloor N \rfloor) \prec P'$
by(*elim inputPermFrame*) *auto*
with S $\langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto M(\lfloor N \rfloor)$
 $\prec P'$
by(*simp add: eqvts*)
moreover from *FrP* **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by simp**
with S $\langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$ *Aeq* **have** *extractFrame* $P = \langle (p \cdot$
 $A_P), p \cdot \Psi_P \rangle$
by(*simp add: eqvts*)
moreover from *FrQ* **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by simp**
with S $\langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$ *Aeq* **have** *extractFrame* $Q = \langle (p \cdot$
 $A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$ **have** *distinct*($p \cdot A_P$) **and** *distinct*(p

$\cdot A_Q$)
by *simp+*
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by** (*simp add: pt-fresh-star-bij* [*OF pt-name-inst, OF at-name-inst*])
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by** (*simp add: pt-fresh-star-bij* [*OF pt-name-inst, OF at-name-inst*])
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by** (*simp add: pt-fresh-star-bij* [*OF pt-name-inst, OF at-name-inst*])
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$ *Aeq Ψ eq*
by (*intro rPar1*) (*assumption* | *simp*)+
next
case (*cPar2* $Q' A_P \Psi_P$)
from $\langle A_P \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_{PQ}$ **by** *simp+*
obtain $A_Q \Psi_Q$ **where** *FrQ: extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **and** *distinct* A_Q
 $A_Q \#* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by (*rule freshFrame*)
then have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* M$ **and** $A_Q \#* A_P$
and $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_P$
by *simp+*

have *FrP: extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ *FrQ* **have** $A_P \#* \Psi_Q$
by (*force dest: extractFrameFreshChain*)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ *FrP FrQ* $\langle A_Q \#* A_P \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_Q \#* A_P \rangle$ **have** *distinct* $(A_P @ A_Q)$
by (*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and *distinctPerm* p
and *Ψ eq:* $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** *Aeq:* $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by (*elim frameChainEq'*) (*assumption* | *simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M(\lfloor N \rfloor) \prec Q' \rangle$ S $\langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle$ *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto M(\lfloor N \rfloor) \prec Q'$
by (*elim inputPermFrame*) *auto*
with S $\langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto M(\lfloor N \rfloor) \prec Q'$
by (*simp add: eqvts*)
moreover from *FrP* **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with S $\langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$ *Aeq* **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P \rangle$
by (*simp add: eqvts*)
moreover from *FrQ* **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*

with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle Aeq$ **have** $extractFrame Q = \langle (p \cdot A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle$ **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_Q \#* A_P \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle Aeq \Psi eq$
by(*intro rPar2*) (*assumption | simp*)+
qed

lemma *parCasesBrInputFrame*[*consumes 7, case-names cPar1 cPar2 cBrMerge*]:

fixes Ψ **::** 'b
and P **::** ('a, 'b, 'c) *psi*
and Q **::** ('a, 'b, 'c) *psi*
and M **::** 'a
and $xvec$ **::** *name list*
and N **::** 'a
and T **::** ('a, 'b, 'c) *psi*
and C **::** 'd::*fs-name*

assumes *Trans*: $\Psi \triangleright P \parallel Q \mapsto_i M(N) \prec T$
and $extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle$
and $distinct A_{PQ}$
and $A_{PQ} \#* \Psi$
and $A_{PQ} \#* P$
and $A_{PQ} \#* Q$
and $A_{PQ} \#* M$
and *rPar1*: $\bigwedge P' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_P; distinct A_Q; A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M; A_P \#* \Psi_Q; A_Q \#* \Psi_P; A_P \#* A_Q; A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \rrbracket \implies Prop (P' \parallel Q)$
and *rPar2*: $\bigwedge Q' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'; extractFrame P = \langle A_P, \Psi_P \rangle; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct A_P; distinct A_Q; A_P \#* \Psi; A_P \#* P; A_P \#* Q; A_P \#* M; A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M; A_P \#* \Psi_Q; A_Q \#* \Psi_P; A_P \#* A_Q; A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \rrbracket \implies Prop (P \parallel Q')$
and *rBrMerge*: $\bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct A_P; \rrbracket$

$\Psi \otimes \Psi_P \triangleright Q \mapsto \iota M(\downarrow N) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle;$
distinct $A_Q;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P;$
 $A_P \#* Q; A_P \#* A_Q;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P;$
 $A_Q \#* Q;$
 $A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies$
 $\text{Prop } (P' \parallel Q')$

shows *Prop T*
using *Trans*
proof(*induct rule: parBrInputCases*[of - - - - - (A_{PQ}, Ψ_{PQ})])
case(*cPar1* $P' A_Q \Psi_Q$)
from $\langle A_Q \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_{PQ}$ **by** *simp+*
obtain $A_P \Psi_P$ **where** *FrP: extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** *distinct* A_P
 $A_P \#* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \#* P$ **and** $A_P \#* Q$ **and** $A_P \#* \Psi$ **and** $A_P \#* M$ **and** $A_P \#* A_Q$
and $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_Q$
by *simp+*

have *FrQ: extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle$ *FrP* **have** $A_Q \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ *FrP FrQ* $\langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** *distinct*($A_P @ A_Q$)
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and *distinctPerm* p
and *Ψeq: Ψ_{PQ} = (p · Ψ_P) ⊗ (p · Ψ_Q)* **and** *Aeq: A_{PQ} = (p · A_P) @ (p · A_Q)*
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption | simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(\downarrow N) \prec P' \rangle$ $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$
 $\langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle$ *Aeq*
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto \iota M(\downarrow N) \prec P'$
by(*elim brinputPermFrame*) *auto*
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ *Aeq* **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto \iota M(\downarrow N) \prec P'$
by(*simp add: eqvts*)

moreover from *FrP* **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$ *Aeq* **have** *extractFrame* $P = \langle (p \cdot A_P), p \cdot \Psi_P \rangle$
by(*simp add: eqvts*)

moreover from *FrQ* **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$ *Aeq* **have** *extractFrame* $Q = \langle (p \cdot$

$A_Q), p \cdot \Psi_Q$
by(*simp add: eqvts*)
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **and** $\text{distinct}(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$ *Aeq Ψ eq*
by(*intro rPar1*) (*assumption | simp*)+
next
case(*cPar2* $Q' A_P \Psi_P$)
from $\langle A_P \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_{PQ}$ **by** *simp+*
obtain $A_Q \Psi_Q$ **where** *FrQ: extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **and** $\text{distinct } A_Q$
 $A_Q \#* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* M$ **and** $A_Q \#* A_P$
and $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_P$
by *simp+*

have *FrP: extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ *FrQ* **have** $A_P \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ *FrP FrQ* $\langle A_Q \#* A_P \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_Q \#* A_P \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and $\text{distinctPerm } p$
and *Ψ eq: $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$* **and** *$A$ eq: $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$*
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption | simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto {}_i M(|N|) \prec Q' \rangle$ $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$
 $\langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle$ *Aeq*
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto {}_i M(|N|) \prec Q'$
by(*elim brinputPermFrame*) *auto*
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ *Aeq* **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto {}_i M(|N|) \prec Q'$
by(*simp add: eqvts*)
moreover from *FrP* **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$ *Aeq* **have** $\text{extractFrame } P = \langle (p \cdot$

A_P), $p \cdot \Psi_P$)
by(*simp add: eqvts*)
moreover from FrQ **have** $(p \cdot extractFrame Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle Aeq$ **have** $extractFrame Q = \langle (p \cdot A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle$ **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_Q \#* A_P \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle Aeq \Psi eq$
by(*intro rPar2*) (*assumption | simp*)+
next
case(*cBrMerge* $\Psi_Q P' A_P \Psi_P Q' A_Q$)
then have $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$
and $A_P \#* A_{PQ}$ **and** $A_Q \#* A_{PQ}$
by *simp+*

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle FrP FrQ \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $distinct(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: set p \subseteq set(A_P @ A_Q) \times set((p \cdot A_P) @ (p \cdot A_Q))$
and $distinctPerm p$
and $\Psi eq: \Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $Aeq: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle distinct A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption | simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P' \rangle S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle Aeq$
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i M(|N|) \prec P'$
by(*elim brinputPermFrame*) *auto*
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle Aeq$ **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto_i M(|N|) \prec P'$
by(*simp add: eqvts*)

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(|N|) \prec Q' \rangle S \langle A_{PQ} \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P \#* Q \rangle \langle A_{PQ} \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* M \rangle Aeq$
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i M(|N|) \prec Q'$
by(*elim brinputPermFrame*) *auto*
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle Aeq$ **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M(|N|)$

$\prec Q'$
by(*simp add: eqvts*)

note $\langle \Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto_{\iota} M(\downarrow N) \prec P' \rangle \langle \Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_{\iota} M(\downarrow N) \prec Q' \rangle$
moreover from *FrP* **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$ *Aeq* **have** $\text{extractFrame } P = \langle (p \cdot A_P), p \cdot \Psi_P \rangle$
by(*simp add: eqvts*)
moreover from *FrQ* **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$ *Aeq* **have** $\text{extractFrame } Q = \langle (p \cdot A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **and** $\text{distinct}(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$ *Aeq* Ψeq
by(*intro rBrMerge*) *simp+*
qed

lemma *parCasesOutputFrame*[*consumes 11, case-names cPar1 cPar2*]:

fixes Ψ **::** 'b
and P **::** ('a, 'b, 'c) *psi*
and Q **::** ('a, 'b, 'c) *psi*
and M **::** 'a
and $xvec$ **::** *name list*
and N **::** 'a
and T **::** ('a, 'b, 'c) *psi*
and C **::** 'd::*fs-name*

assumes *Trans*: $\Psi \triangleright P \parallel Q \mapsto M(\downarrow \nu * xvec) \langle N \rangle \prec T$

and $xvec \#* \Psi$
and $xvec \#* P$
and $xvec \#* Q$
and $xvec \#* M$
and $\text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle$
and $\text{distinct } A_{PQ}$
and $A_{PQ} \#* \Psi$
and $A_{PQ} \#* P$
and $A_{PQ} \#* Q$
and $A_{PQ} \#* M$
and *rPar1*: $\bigwedge P' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\downarrow \nu * xvec) \langle N \rangle \prec P' \rrbracket$

$extractFrame P = \langle A_P, \Psi_P \rangle$; $extractFrame Q = \langle A_Q, \Psi_Q \rangle$;
 $distinct A_P$; $distinct A_Q$; $A_P \#* \Psi$; $A_P \#* P$; $A_P \#*$
 Q ; $A_P \#* M$; $A_Q \#* \Psi$; $A_Q \#* P$; $A_Q \#* Q$; $A_Q \#* M$;
 $A_P \#* \Psi_Q$; $A_Q \#* \Psi_P$; $A_P \#* A_Q$; $A_{PQ} = A_P @ A_Q$;
 $\Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies Prop (P' \parallel Q)$
and $rPar2: \bigwedge Q' A_P \Psi_P A_Q \Psi_Q. [\Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu * xvec)\langle N \rangle \prec Q']$;
 $extractFrame P = \langle A_P, \Psi_P \rangle$; $extractFrame Q = \langle A_Q, \Psi_Q \rangle$;
 $distinct A_P$; $distinct A_Q$; $A_P \#* \Psi$; $A_P \#* P$; $A_P \#*$
 Q ; $A_P \#* M$; $A_Q \#* \Psi$; $A_Q \#* P$; $A_Q \#* Q$; $A_Q \#* M$;
 $A_P \#* \Psi_Q$; $A_Q \#* \Psi_P$; $A_P \#* A_Q$; $A_{PQ} = A_P @ A_Q$;
 $\Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies Prop (P \parallel Q')$
shows $Prop T$
using $Trans \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle xvec \#* M \rangle$
proof(*induct rule: parOutputCases*[of - - - - - (A_{PQ}, Ψ_{PQ})])
case($cPar1 P' A_Q \Psi_Q$)
from $\langle A_Q \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_{PQ}$ **by** $simp+$
obtain $A_P \Psi_P$ **where** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $distinct A_P$
 $A_P \#* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by(*rule freshFrame*)
then have $A_P \#* P$ **and** $A_P \#* Q$ **and** $A_P \#* \Psi$ **and** $A_P \#* M$ **and** $A_P \#* A_Q$
and $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_Q$
by $simp+$

have $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle FrP$ **have** $A_Q \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle FrP FrQ \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** $simp$
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $distinct(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: set p \subseteq set(A_P @ A_Q) \times set((p \cdot A_P) @ (p \cdot A_Q))$
and $distinctPerm p$
and $\Psi_{PQ}: \Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $A_{PQ}: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle distinct A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption* | *simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P' \rangle S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#*$
 $P \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle A_{PQ}$
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'$
by(*elim outputPermFrame*) (*assumption* | *simp*)+

with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle A_{PQ}$ **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P$
 $\mapsto M(\nu * xvec)\langle N \rangle \prec P'$
by(*simp add: eqvts*)
moreover from FrP **have** $(p \cdot extractFrame P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** $simp$
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle A_{PQ}$ **have** $extractFrame P = \langle (p \cdot$

$A_P), p \cdot \Psi_P)$
by(*simp add: eqvts*)
moreover from FrQ **have** $(p \cdot extractFrame Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$ Aeq **have** $extractFrame Q = \langle (p \cdot A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle$ **have** $distinct(p \cdot A_P)$ **and** $distinct(p \cdot A_Q)$
by *simp+*
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case using* $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$ $Aeq \Psi eq$
by(*intro rPar1*) (*assumption | simp*)+
next
case(*cPar2 Q' A_P \Psi_P*)
from $\langle A_P \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_{PQ}$ **by** *simp+*
obtain $A_Q \Psi_Q$ **where** $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** $distinct A_Q$
 $A_Q \#* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by(*rule freshFrame*)
then have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* M$ **and** $A_Q \#* A_P$ **and** $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_P$
by *simp+*

have $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ FrQ **have** $A_P \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ $FrP FrQ \langle A_Q \#* A_P \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_Q \#* A_P \rangle$ **have** $distinct(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** $S: set p \subseteq set(A_P @ A_Q) \times set((p \cdot A_P) @ (p \cdot A_Q))$
and $distinctPerm p$
and $\Psi eq: \Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $Aeq: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle distinct A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption | simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec Q' \rangle$ $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle$ Aeq
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec Q'$
by(*elim outputPermFrame*) (*assumption | simp*)+
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ Aeq **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q$

```

 $\mapsto M(\nu*xvec)\langle N \rangle \prec Q'$ 
  by(simp add: eqvts)
  moreover from FrP have  $(p \cdot extractFrame P) = p \cdot \langle A_P, \Psi_P \rangle$  by simp
  with  $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$  Aeq have  $extractFrame P = \langle (p \cdot A_P), p \cdot \Psi_P \rangle$ 
  by(simp add: eqvts)
  moreover from FrQ have  $(p \cdot extractFrame Q) = p \cdot \langle A_Q, \Psi_Q \rangle$  by simp
  with  $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$  Aeq have  $extractFrame Q = \langle (p \cdot A_Q), p \cdot \Psi_Q \rangle$ 
  by(simp add: eqvts)
  moreover from  $\langle distinct A_P \rangle \langle distinct A_Q \rangle$  have  $distinct(p \cdot A_P)$  and  $distinct(p \cdot A_Q)$ 
  by simp+
  moreover from  $\langle A_Q \#* A_P \rangle$  have  $(p \cdot A_P) \#* (p \cdot A_Q)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  moreover from  $\langle A_P \#* \Psi_Q \rangle$  have  $(p \cdot A_P) \#* (p \cdot \Psi_Q)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  moreover from  $\langle A_Q \#* \Psi_P \rangle$  have  $(p \cdot A_Q) \#* (p \cdot \Psi_P)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
  ultimately show ?case using  $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$  Aeq Ψeq
  by(intro rPar2) (assumption | simp)+
qed

```

lemma *parCasesBrOutputFrame*[*consumes 11*, *case-names cPar1 cPar2 cBrComm1 cBrComm2*]:

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $Q$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $xvec$   :: name list
and  $N$      :: 'a
and  $T$      :: ('a, 'b, 'c) psi
and  $C$      :: 'd::fs-name

```

```

assumes Trans:  $\Psi \triangleright P \parallel Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec T$ 
and  $xvec \#* \Psi$ 
and  $xvec \#* P$ 
and  $xvec \#* Q$ 
and  $xvec \#* M$ 
and  $extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle$ 
and  $distinct A_{PQ}$ 
and  $A_{PQ} \#* \Psi$ 
and  $A_{PQ} \#* P$ 
and  $A_{PQ} \#* Q$ 
and  $A_{PQ} \#* M$ 
and rPar1:  $\bigwedge P' A_P \Psi_P A_Q \Psi_Q. \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P' \rrbracket$ ;
 $extractFrame P = \langle A_P, \Psi_P \rangle$ ;  $extractFrame Q = \langle A_Q, \Psi_Q \rangle$ ;
 $distinct A_P$ ;  $distinct A_Q$ ;  $A_P \#* \Psi$ ;  $A_P \#* P$ ;  $A_P \#*$ 

```

$Q; A_P \#* M; A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M;$
 $A_P \#* \Psi_Q; A_Q \#* \Psi_P; A_P \#* A_Q; A_{PQ} = A_P @ A_Q;$
 $\Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies Prop (P' \parallel Q)$
and $rPar2: \bigwedge Q' A_P \Psi_P A_Q \Psi_Q. [\Psi \otimes \Psi_P \triangleright Q \mapsto_{iM}(\nu * xvec)\langle N \rangle \prec Q';$
 $extractFrame P = \langle A_P, \Psi_P \rangle; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_P; distinct A_Q; A_P \#* \Psi; A_P \#* P; A_P \#*$
 $Q; A_P \#* M; A_Q \#* \Psi; A_Q \#* P; A_Q \#* Q; A_Q \#* M;$
 $A_P \#* \Psi_Q; A_Q \#* \Psi_P; A_P \#* A_Q; A_{PQ} = A_P @ A_Q;$
 $\Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies Prop (P \parallel Q')$
and $rBrComm1: \bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM}(\nu * xvec)\langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle; distinct$
 $A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_{iM}(\nu * xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle;$
 $distinct A_Q;$
 $distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* Q; A_P \#* A_Q;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* Q;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* Q;$
 $A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies$
 $Prop (P' \parallel Q')$
and $rBrComm2: \bigwedge \Psi_Q P' A_P \Psi_P Q' A_Q.$
 $[\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM}(\nu * xvec)\langle N \rangle \prec P'; extractFrame P = \langle A_P, \Psi_P \rangle;$
 $distinct A_P;$
 $\Psi \otimes \Psi_P \triangleright Q \mapsto_{iM}(\nu * xvec)\langle N \rangle \prec Q'; extractFrame Q = \langle A_Q, \Psi_Q \rangle; distinct$
 $A_Q;$
 $distinct xvec;$
 $A_P \#* \Psi; A_P \#* \Psi_Q; A_P \#* P; A_P \#* Q; A_P \#* A_Q;$
 $A_Q \#* \Psi; A_Q \#* \Psi_P; A_Q \#* P; A_Q \#* Q;$
 $A_P \#* M; A_Q \#* M; xvec \#* M;$
 $xvec \#* \Psi; xvec \#* P; xvec \#* Q;$
 $A_{PQ} = A_P @ A_Q; \Psi_{PQ} = \Psi_P \otimes \Psi_Q \implies$
 $Prop (P' \parallel Q')$

shows Prop T
using $Trans \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle xvec \#* M \rangle$
proof($induct\ rule: parBrOutputCases[of\ \text{-----}\ (A_{PQ}, \Psi_{PQ})]$)
case($cPar1 P' A_Q \Psi_Q$)
from $\langle A_Q \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_{PQ}$ **by** $simp+$
obtain $A_P \Psi_P$ **where** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $distinct A_P$
 $A_P \#* (P, Q, \Psi, M, A_Q, A_{PQ}, \Psi_Q)$
by($rule\ freshFrame$)
then have $A_P \#* P$ **and** $A_P \#* Q$ **and** $A_P \#* \Psi$ **and** $A_P \#* M$ **and** $A_P \#* A_Q$
and $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_Q$
by $simp+$

have $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **by** $fact$

from $\langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle$ FrP **have** $A_Q \#* \Psi_P$
by($force\ dest: extractFrameFreshChain$)

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \text{FrP FrQ} \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by *(auto simp add: fresh-star-def fresh-def name-list-supp)*
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and $\text{distinctPerm } p$
and $\Psi_{PQ}: \Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $\text{Aeq}: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by *(elim frameChainEq') (assumption | simp add: eqvts)+*

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_{\text{jM}} (\nu * \text{xvec}) \langle N \rangle \prec P' \rangle S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#*$
 $P \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \text{Aeq}$
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_{\text{jM}} (\nu * \text{xvec}) \langle N \rangle \prec P'$
by *(elim brotputPermFrame) (assumption | simp)+*

with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \text{Aeq}$ **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P$
 $\mapsto_{\text{jM}} (\nu * \text{xvec}) \langle N \rangle \prec P'$
by *(simp add: eqvts)*
moreover from FrP **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \text{Aeq}$ **have** $\text{extractFrame } P = \langle (p \cdot$
 $A_P), p \cdot \Psi_P \rangle$
by *(simp add: eqvts)*
moreover from FrQ **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \text{Aeq}$ **have** $\text{extractFrame } Q = \langle (p \cdot$
 $A_Q), p \cdot \Psi_Q \rangle$
by *(simp add: eqvts)*
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **and** $\text{distinct}(p$
 $\cdot A_Q)$
by *simp+*
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by** *(simp add: pt-fresh-star-bij[OF*
 $\text{pt-name-inst, OF at-name-inst}]$
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by** *(simp add: pt-fresh-star-bij[OF*
 $\text{pt-name-inst, OF at-name-inst}]$
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by** *(simp add: pt-fresh-star-bij[OF*
 $\text{pt-name-inst, OF at-name-inst}]$
ultimately show $?case$ **using** $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#*$
 $M \rangle \text{Aeq } \Psi_{PQ}$
by *(intro rPar1) (assumption | simp)+*
next
case *(cPar2 Q' A_P Ψ_P)*
from $\langle A_P \#* (A_{PQ}, \Psi_{PQ}) \rangle$ **have** $A_P \#* A_{PQ}$ **and** $A_P \#* \Psi_{PQ}$ **by** *simp+*
obtain $A_Q \Psi_Q$ **where** $\text{FrQ}: \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **and** $\text{distinct } A_Q$
 $A_Q \#* (P, Q, \Psi, M, A_P, A_{PQ}, \Psi_P)$
by *(rule freshFrame)*
then have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* M$ **and** $A_Q \#* A_P$
and $A_Q \#* A_{PQ}$ **and** $A_Q \#* \Psi_P$
by *simp+*

have FrP : $extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact*

from $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle FrQ$ **have** $A_P \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

from $\langle extractFrame(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle FrP FrQ \langle A_Q \#* A_P \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle \langle A_Q \#* A_P \rangle$ **have** $distinct(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S : $set p \subseteq set(A_P @ A_Q) \times set((p \cdot A_P) @ (p \cdot A_Q))$
and $distinctPerm p$
and $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle distinct A_{PQ} \rangle$
by(*elim frameChainEq' (assumption | simp add: eqvts)+*)

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_{jM} (\nu * xvec) \langle N \rangle \prec Q' \rangle S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q$
 $\#* Q \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle A_{eq}$
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_{jM} (\nu * xvec) \langle N \rangle \prec Q'$
by(*elim brooutputPermFrame (assumption | simp)+*)
with $S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle A_{eq}$ **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q$
 $\mapsto_{jM} (\nu * xvec) \langle N \rangle \prec Q'$
by(*simp add: eqvts*)
moreover from FrP **have** $(p \cdot extractFrame P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle A_{eq}$ **have** $extractFrame P = \langle (p \cdot$
 $A_P), p \cdot \Psi_P \rangle$
by(*simp add: eqvts*)
moreover from FrQ **have** $(p \cdot extractFrame Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle A_{eq}$ **have** $extractFrame Q = \langle (p \cdot$
 $A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle distinct A_P \rangle \langle distinct A_Q \rangle$ **have** $distinct(p \cdot A_P)$ **and** $distinct(p$
 $\cdot A_Q)$
by *simp+*
moreover from $\langle A_Q \#* A_P \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF*
 $pt-name-inst, OF at-name-inst]$)
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF*
 $pt-name-inst, OF at-name-inst]$)
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF*
 $pt-name-inst, OF at-name-inst]$)
ultimately show $?case$ **using** $\langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#*$
 $M \rangle A_{eq} \Psi_{eq}$
by(*intro rPar2 (assumption | simp)+*)
next
case(*cBrComm1* $\Psi_Q P' A_P \Psi_P Q' A_Q$)
then have FrP : $extractFrame P = \langle A_P, \Psi_P \rangle$ **and** FrQ : $extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle$
and $A_P \#* A_{PQ}$ **and** $A_Q \#* A_{PQ}$

by simp+

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle \text{FrP FrQ} \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by simp**
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle \langle A_P \#* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by $(\text{auto simp add: fresh-star-def fresh-def name-list-supp})$
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(A_P @ A_Q) \times \text{set}((p \cdot A_P) @ (p \cdot A_Q))$
and $\text{distinctPerm } p$
and $\Psi_{PQ}: \Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** $\text{Aeq}: A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle \langle A_Q \#* A_{PQ} \rangle \langle \text{distinct } A_{PQ} \rangle$
by $(\text{elim frameChainEq}') (\text{assumption} \mid \text{simp add: eqvts})+$

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P' \rangle S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle$
 $\langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \text{Aeq}$
have $\langle p \cdot (\Psi \otimes \Psi_Q) \triangleright P \mapsto_i M(N) \prec P' \rangle$
by $(\text{elim brinputPermFrame}) (\text{assumption} \mid \text{simp})+$

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * \text{xvec})(N) \prec Q' \rangle S \langle A_{PQ} \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P$
 $\#* Q \rangle \langle A_{PQ} \#* M \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \text{Aeq}$
have $\langle p \cdot (\Psi \otimes \Psi_P) \triangleright Q \mapsto_i M(\nu * \text{xvec})(N) \prec Q' \rangle$
by $(\text{elim brotputPermFrame}) (\text{assumption} \mid \text{simp})+$

from $\langle (p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i M(N) \prec P' \rangle S \langle A_{PQ} \#* \Psi \rangle \langle A_P \#* \Psi \rangle \langle A_Q \#*$
 $\Psi \rangle \text{Aeq}$ **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto_i M(N) \prec P'$
by (simp add: eqvts)
moreover from $\langle (p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i M(\nu * \text{xvec})(N) \prec Q' \rangle S \langle A_{PQ} \#* \Psi \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \text{Aeq}$ **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M(\nu * \text{xvec})(N) \prec Q'$
by (simp add: eqvts)

moreover from FrP **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by simp**
with $S \langle A_{PQ} \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \text{Aeq}$ **have** $\text{extractFrame } P = \langle (p \cdot$
 $A_P), p \cdot \Psi_P \rangle$
by (simp add: eqvts)
moreover from FrQ **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by simp**
with $S \langle A_{PQ} \#* Q \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \text{Aeq}$ **have** $\text{extractFrame } Q = \langle (p \cdot$
 $A_Q), p \cdot \Psi_Q \rangle$
by (simp add: eqvts)
moreover from $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **and** $\text{distinct}(p$
 $\cdot A_Q)$
by simp+
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by** $(\text{simp add: pt-fresh-star-bij}[OF$
 $\text{pt-name-inst, OF at-name-inst}])$
moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by** $(\text{simp add: pt-fresh-star-bij}[OF$
 $\text{pt-name-inst, OF at-name-inst}])$
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by** $(\text{simp add: pt-fresh-star-bij}[OF$
 $\text{pt-name-inst, OF at-name-inst}])$
ultimately show $?case$ **using** $\langle \text{distinct } \text{xvec} \rangle \langle \text{xvec} \#* M \rangle \langle \text{xvec} \#* \Psi \rangle \langle \text{xvec} \#*$
 $P \rangle \langle \text{xvec} \#* Q \rangle \langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle \text{Aeq } \Psi_{PQ}$

by(*intro rBrComm1*) (*assumption* | *simp*)+
next
case(*cBrComm2* Ψ_Q P' A_P Ψ_P Q' A_Q)
then have FrP : *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** FrQ : *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
and $A_P \#* A_{PQ}$ **and** $A_Q \#* A_{PQ}$
by *simp*+

from $\langle \text{extractFrame}(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ} \rangle \rangle$ FrP FrQ $\langle A_P \#* A_Q \rangle$ $\langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle = \langle A_{PQ}, \Psi_{PQ} \rangle$ **by** *simp*
moreover from $\langle \text{distinct } A_P \rangle$ $\langle \text{distinct } A_Q \rangle$ $\langle A_P \#* A_Q \rangle$ **have** $\text{distinct}(A_P @ A_Q)$
by(*auto simp add: fresh-star-def fresh-def name-list-supp*)
ultimately obtain p **where** S : $set\ p \subseteq set(A_P @ A_Q) \times set((p \cdot A_P) @ (p \cdot A_Q))$
and $\text{distinctPerm } p$
and Ψ_{PQ} : $\Psi_{PQ} = (p \cdot \Psi_P) \otimes (p \cdot \Psi_Q)$ **and** A_{PQ} : $A_{PQ} = (p \cdot A_P) @ (p \cdot A_Q)$
using $\langle A_P \#* A_{PQ} \rangle$ $\langle A_Q \#* A_{PQ} \rangle$ $\langle \text{distinct } A_{PQ} \rangle$
by(*elim frameChainEq'*) (*assumption* | *simp add: eqvts*)+

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P' \rangle$ S $\langle A_{PQ} \#* P \rangle$ $\langle A_P \#* P \rangle$ $\langle A_Q \#*$
 $P \rangle$ $\langle A_{PQ} \#* M \rangle$ $\langle A_P \#* M \rangle$ $\langle A_Q \#* M \rangle$ A_{eq}
have $(p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'$
by(*elim brotputPermFrame*) (*assumption* | *simp*)+

from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q' \rangle$ S $\langle A_{PQ} \#* Q \rangle$ $\langle A_Q \#* Q \rangle$ $\langle A_P \#* Q \rangle$
 $\langle A_{PQ} \#* M \rangle$ $\langle A_P \#* M \rangle$ $\langle A_Q \#* M \rangle$ A_{eq}
have $(p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'$
by(*elim brinputPermFrame*) (*assumption* | *simp*)+

from $\langle (p \cdot (\Psi \otimes \Psi_Q)) \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P' \rangle$ S $\langle A_{PQ} \#* \Psi \rangle$ $\langle A_P \#* \Psi \rangle$
 $\langle A_Q \#* \Psi \rangle$ A_{eq} **have** $\Psi \otimes (p \cdot \Psi_Q) \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'$
by(*simp add: eqvts*)
moreover from $\langle (p \cdot (\Psi \otimes \Psi_P)) \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q' \rangle$ S $\langle A_{PQ} \#* \Psi \rangle$ $\langle A_P \#*$
 $\Psi \rangle$ $\langle A_Q \#* \Psi \rangle$ A_{eq} **have** $\Psi \otimes (p \cdot \Psi_P) \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'$
by(*simp add: eqvts*)

moreover from FrP **have** $(p \cdot \text{extractFrame } P) = p \cdot \langle A_P, \Psi_P \rangle$ **by** *simp*
with S $\langle A_{PQ} \#* P \rangle$ $\langle A_P \#* P \rangle$ $\langle A_Q \#* P \rangle$ A_{eq} **have** $\text{extractFrame } P = \langle (p \cdot$
 $A_P), p \cdot \Psi_P \rangle$
by(*simp add: eqvts*)
moreover from FrQ **have** $(p \cdot \text{extractFrame } Q) = p \cdot \langle A_Q, \Psi_Q \rangle$ **by** *simp*
with S $\langle A_{PQ} \#* Q \rangle$ $\langle A_P \#* Q \rangle$ $\langle A_Q \#* Q \rangle$ A_{eq} **have** $\text{extractFrame } Q = \langle (p \cdot$
 $A_Q), p \cdot \Psi_Q \rangle$
by(*simp add: eqvts*)
moreover from $\langle \text{distinct } A_P \rangle$ $\langle \text{distinct } A_Q \rangle$ **have** $\text{distinct}(p \cdot A_P)$ **and** $\text{distinct}(p$
 $\cdot A_Q)$
by *simp*+
moreover from $\langle A_P \#* A_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot A_Q)$ **by**(*simp add: pt-fresh-star-bij[OF*
pt-name-inst, OF at-name-inst])

moreover from $\langle A_P \#* \Psi_Q \rangle$ **have** $(p \cdot A_P) \#* (p \cdot \Psi_Q)$ **by** (*simp add: pt-fresh-star-bij* [*OF pt-name-inst, OF at-name-inst*])
moreover from $\langle A_Q \#* \Psi_P \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot \Psi_P)$ **by** (*simp add: pt-fresh-star-bij* [*OF pt-name-inst, OF at-name-inst*])
ultimately show *?case using* $\langle \text{distinct } xvec \rangle \langle xvec \#* M \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle A_{PQ} \#* \Psi \rangle \langle A_{PQ} \#* P \rangle \langle A_{PQ} \#* Q \rangle \langle A_{PQ} \#* M \rangle$ *Aeq* Ψeq
by (*intro rBrComm2*) (*assumption* | *simp*)
qed

inductive *bangPred* :: $('a, 'b, 'c) \text{ psi} \Rightarrow ('a, 'b, 'c) \text{ psi} \Rightarrow \text{bool}$
where
aux1: *bangPred* *P* (!*P*)
| *aux2*: *bangPred* *P* (*P* || !*P*)

lemma *bangInduct* [*consumes 1, case-names cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBang*]:

fixes Ψ :: $'b$
and *P* :: $('a, 'b, 'c) \text{ psi}$
and *Rs* :: $('a, 'b, 'c) \text{ residual}$
and *Prop* :: $'d :: \text{fs-name} \Rightarrow 'b \Rightarrow ('a, 'b, 'c) \text{ psi} \Rightarrow ('a, 'b, 'c) \text{ residual} \Rightarrow \text{bool}$
and *C* :: $'d$

assumes $\Psi \triangleright !P \mapsto Rs$

and *rPar1*: $\bigwedge \alpha P' C. [\Psi \triangleright P \mapsto \alpha \prec P'; \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P; \text{bn } \alpha \#* \text{subject } \alpha; \text{bn } \alpha \#* C; \text{distinct}(\text{bn } \alpha)] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\alpha \prec (P' \parallel !P))$
and *rPar2*: $\bigwedge \alpha P' C. [\Psi \triangleright !P \mapsto \alpha \prec P'; \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P; \text{bn } \alpha \#* \text{subject } \alpha; \text{bn } \alpha \#* C; \text{distinct}(\text{bn } \alpha); \bigwedge C. \text{Prop } C \Psi (!P) (\alpha \prec P')] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\alpha \prec (P \parallel P'))$
and *rComm1*: $\bigwedge M N P' K xvec P'' C. [\Psi \triangleright P \mapsto M(\downarrow N) \prec P'; \Psi \triangleright !P \mapsto K(\downarrow \nu * xvec) \langle N \rangle \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (K(\downarrow \nu * xvec) \langle N \rangle \prec P''); \Psi \vdash M \leftrightarrow K; xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* C; \text{distinct } xvec] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\tau \prec (\downarrow \nu * xvec) \langle P' \parallel P'' \rangle)$
and *rComm2*: $\bigwedge M xvec N P' K P'' C. [\Psi \triangleright P \mapsto M(\downarrow \nu * xvec) \langle N \rangle \prec P'; \Psi \triangleright !P \mapsto K(\downarrow N) \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (K(\downarrow N) \prec P''); \Psi \vdash M \leftrightarrow K; xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* C; \text{distinct } xvec] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\tau \prec (\downarrow \nu * xvec) \langle P' \parallel P'' \rangle)$
and *rBrMerge*: $\bigwedge M N P' P'' C. [\Psi \triangleright P \mapsto \downarrow M(\downarrow N) \prec P'; \Psi \triangleright !P \mapsto \downarrow M(\downarrow N) \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (\downarrow M(\downarrow N) \prec P'')] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\downarrow M(\downarrow N) \prec (P' \parallel P''))$
and *rBrComm1*: $\bigwedge M N P' xvec P'' C. [\Psi \triangleright P \mapsto \downarrow M(\downarrow N) \prec P'; \Psi \triangleright !P \mapsto \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (\downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P''); xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; \text{distinct } xvec] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec (P' \parallel P''))$
and *rBrComm2*: $\bigwedge M N P' xvec P'' C. [\Psi \triangleright P \mapsto \downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec P'; \Psi \triangleright !P \mapsto \downarrow M(\downarrow N) \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (\downarrow M(\downarrow N) \prec P''); xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; \text{distinct } xvec] \Longrightarrow \text{Prop } C \Psi (P \parallel !P) (\downarrow M(\downarrow \nu * xvec) \langle N \rangle \prec (P' \parallel P''))$

```

and rBang:  $\bigwedge R s C. \llbracket \Psi \triangleright P \parallel !P \mapsto R s; \bigwedge C. Prop C \Psi (P \parallel !P) R s; guarded P \rrbracket \implies Prop C \Psi (!P) R s$ 
shows Prop C  $\Psi (!P) R s$ 
proof –
  from  $\langle \Psi \triangleright !P \mapsto R s \rangle$  have guarded P
  by(nominal-induct  $\Psi P == !P R s$  rule: semantics.strong-induct) (auto simp add: psi.inject)
  {
    fix Q :: ('a, 'b, 'c) psi
    and  $\Psi' :: 'b$ 

    assume  $\Psi' \triangleright Q \mapsto R s$ 
    and guarded Q
    and bangPred P Q
    and  $\Psi \simeq \Psi'$ 

    then have Prop C  $\Psi Q R s$  using rPar1 rPar1 rPar2 rPar2 rComm1 rComm2
    rBrMerge rBrComm1 rBrComm2 rBang
    proof(nominal-induct avoiding:  $\Psi C$  rule: semantics.strong-induct)
      case(cInput  $\Psi' M K xvec N Tvec Q \Psi C$ )
        then show ?case by – (ind-cases bangPred P (M( $\lambda * xvec N$ )).Q))
      next
        case(cBrInput  $\Psi' K M xvec N Tvec Q \Psi C$ )
          then show ?case by – (ind-cases bangPred P (M( $\lambda * xvec N$ )).Q))
        next
          case(Output  $\Psi' M K N Q \Psi C$ )
            then show ?case by – (ind-cases bangPred P (M( $\langle N \rangle$ )).Q))
          next
            case(BrOutput  $\Psi' M K N Q \Psi C$ )
              then show ?case by – (ind-cases bangPred P (M( $\langle N \rangle$ )).Q))
            next
              case(Case  $\Psi' Q R s \varphi C s \Psi C$ )
                then show ?case by – (ind-cases bangPred P (Cases Cs))
              next
                case(cPar1  $\Psi' \Psi_R Q \alpha P' R A_R \Psi C$ )
                  have rPar1:  $\bigwedge \alpha P' C. \llbracket \Psi \triangleright P \mapsto \alpha \prec P'; bn \alpha \#* \Psi; bn \alpha \#* P; bn \alpha \#* subject \alpha; bn \alpha \#* C; distinct(bn \alpha) \rrbracket \implies Prop C \Psi (P \parallel !P) (\alpha \prec (P' \parallel !P))$ 
                  by fact
                  from  $\langle bangPred P (Q \parallel R) \rangle$  have Q = P and R = !P
                  by – (ind-cases bangPred P (Q  $\parallel$  R), auto simp add: psi.inject)+
                  from  $\langle R = !P \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$  have  $A_R = []$  and  $\Psi_R = \mathbf{1}$  by
auto
                  from  $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto \alpha \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$  have  $\Psi \triangleright P \mapsto \alpha \prec P'$ 
                  by(metis statEqTransition Identity AssertionStatEqSym)
                  then have Prop C  $\Psi (P \parallel !P) (\alpha \prec (P' \parallel !P))$  using  $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* C \rangle \langle Q = P \rangle \langle distinct(bn \alpha) \rangle$ 
                  by(intro rPar1) auto
                  with  $\langle R = !P \rangle \langle Q = P \rangle$  show ?case by simp

```

```

next
  case(cPar2  $\Psi' \Psi_P R \alpha P' Q A_P \Psi C$ )
  have rPar2:  $\bigwedge \alpha P' C. \llbracket \Psi \triangleright !P \mapsto \alpha \prec P'; \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P; \text{bn } \alpha \#* \text{subject } \alpha; \text{bn } \alpha \#* C; \text{distinct}(\text{bn } \alpha);$ 
     $\bigwedge C. \text{Prop } C \Psi (!P) (\alpha \prec P') \rrbracket \implies \text{Prop } C \Psi (P \parallel !P) (\alpha$ 
 $\prec (P \parallel P'))$ 
    by fact
  from  $\langle \text{bangPred } P (Q \parallel R) \rangle$  have  $Q = P$  and  $R = !P$ 
  by  $-(\text{ind-cases } \text{bangPred } P (Q \parallel R), \text{auto simp add: psi.inject})+$ 
  from  $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$  have  $\Psi_P \simeq \mathbf{1}$  and
  supp  $\Psi_P = (\{\}::\text{name set})$ 
  by(blast dest: guardedStatEq)+
  from  $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto \alpha \prec P' \rangle \langle R = !P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \triangleright$ 
 $!P \mapsto \alpha \prec P'$ 
  by(metis statEqTransition Identity Composition Commutativity Assertion-
  StatEqSym)
  moreover
  {
  fix C
  have bangPred P (!P) by(rule aux1)
  moreover from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \simeq \Psi' \otimes \Psi_P$  by(metis
  Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
  ultimately have  $\bigwedge C. \text{Prop } C \Psi (!P) (\alpha \prec P')$  using cPar2  $\langle R = !P \rangle$ 
 $\langle \text{guarded } P \rangle$  by simp
  }
  ultimately have  $\text{Prop } C \Psi (P \parallel !P) (\alpha \prec (P \parallel P'))$  using  $\langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn}$ 
 $\alpha \#* Q \rangle \langle \text{bn } \alpha \#* \text{subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle \langle Q = P \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$ 
  by(elim rPar2) auto
  with  $\langle R = !P \rangle \langle Q = P \rangle$  show ?case by simp
next
  case(cComm1  $\Psi' \Psi_R Q M N P' A_P \Psi_P R K \text{vec } P'' A_R \Psi C$ )
  have rComm1:  $\bigwedge M N P' K \text{vec } P'' C. \llbracket \Psi \triangleright P \mapsto M(\downarrow N) \prec P'; \Psi \triangleright !P$ 
 $\mapsto K(\downarrow \nu * \text{vec}) \langle N \rangle \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (K(\downarrow \nu * \text{vec}) \langle N \rangle \prec P''); \Psi \vdash M \leftrightarrow$ 
 $K;$ 
     $\text{vec } \#* \Psi; \text{vec } \#* P; \text{vec } \#* M; \text{vec } \#* K; \text{vec}$ 
 $\#* C; \text{distinct } \text{vec} \rrbracket \implies \text{Prop } C \Psi (P \parallel !P) (\tau \prec (\downarrow \nu * \text{vec}) \langle P' \parallel P'' \rangle)$ 
    by fact
  from  $\langle \text{bangPred } P (Q \parallel R) \rangle$  have  $Q = P$  and  $R = !P$ 
  by  $-(\text{ind-cases } \text{bangPred } P (Q \parallel R), \text{auto simp add: psi.inject})+$ 
  from  $\langle R = !P \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$  have  $A_R = []$  and  $\Psi_R = \mathbf{1}$  by
  auto
  from  $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto M(\downarrow N) \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$  have  $\Psi$ 
 $\triangleright P \mapsto M(\downarrow N) \prec P'$ 
  by(metis statEqTransition Identity AssertionStatEqSym)
  moreover from  $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$  have
   $\Psi_P \simeq \mathbf{1}$  and supp  $\Psi_P = (\{\}::\text{name set})$ 
  by(blast dest: guardedStatEq)+
  moreover from  $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto K(\downarrow \nu * \text{vec}) \langle N \rangle \prec P'' \rangle \langle R = !P \rangle \langle \Psi_P \simeq$ 
 $\mathbf{1} \rangle \langle \Psi \simeq \Psi' \rangle$  have  $\Psi \triangleright !P \mapsto K(\downarrow \nu * \text{vec}) \langle N \rangle \prec P''$ 

```

```

    by(metis statEqTransition Identity Composition Commutativity Assertion-
StatEqSym)
  moreover
  {
    fix C
    have bangPred P (!P) by(rule aux1)
      moreover from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \simeq \Psi' \otimes \Psi_P$  by(metis
Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
      ultimately have  $\bigwedge C. \text{Prop } C \Psi (!P) (K(\nu * xvec) \langle N \rangle \prec P'')$  using cComm1
 $\langle R = !P \rangle \langle \text{guarded } P \rangle$  by simp
    }
  moreover from  $\langle \Psi' \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K \rangle \langle \Psi_P \simeq \mathbf{1} \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$ 
have  $\Psi \vdash M \leftrightarrow K$ 
    by(metis statEqEnt Identity Composition Commutativity AssertionStatE-
qSym)
  ultimately have Prop C Psi (P || !P) (tau < (nu * xvec) (P' || P'')) using  $\langle xvec$ 
 $\#* \Psi \rangle \langle xvec \#* Q \rangle \langle xvec \#* M \rangle \langle xvec \#* K \rangle \langle xvec \#* C \rangle \langle Q = P \rangle \langle \text{distinct } xvec \rangle$ 
    by(elim rComm1 [where K=K and M=M and N=N]) auto
    with  $\langle R = !P \rangle \langle Q = P \rangle$  show ?case by simp
  next
  case(cComm2 Psi' Psi_R Q M xvec N P' A_P Psi_P R K P'' A_R Psi C)
  have rComm2:  $\bigwedge M xvec N P' K P'' C. [\Psi \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'; \Psi$ 
 $\triangleright !P \mapsto K \langle N \rangle \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (K \langle N \rangle \prec P''); \Psi \vdash M \leftrightarrow K;$ 
 $xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* K; xvec$ 
 $\#* C; \text{distinct } xvec] \implies \text{Prop } C \Psi (P || !P) (\tau < (nu * xvec) (P' || P''))$ 
    by fact
  from  $\langle \text{bangPred } P (Q || R) \rangle$  have  $Q = P$  and  $R = !P$ 
    by - (ind-cases bangPred P (Q || R), auto simp add: psi.inject) +
  from  $\langle R = !P \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$  have  $A_R = []$  and  $\Psi_R = \mathbf{1}$  by
auto
  from  $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$ 
have  $\Psi \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'$ 
    by(metis statEqTransition Identity AssertionStatEqSym)
  moreover from  $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$  have
 $\Psi_P \simeq \mathbf{1}$  and supp  $\Psi_P = (\{\} :: \text{name set})$ 
    by(blast dest: guardedStatEq) +
  moreover from  $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto K \langle N \rangle \prec P'' \rangle \langle R = !P \rangle \langle \Psi_P \simeq \mathbf{1} \rangle \langle \Psi \simeq$ 
 $\Psi' \rangle$  have  $\Psi \triangleright !P \mapsto K \langle N \rangle \prec P''$ 
    by(metis statEqTransition Identity Composition Commutativity Assertion-
StatEqSym)
  moreover
  {
    fix C
    have bangPred P (!P) by(rule aux1)
      moreover from  $\langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$  have  $\Psi \simeq \Psi' \otimes \Psi_P$  by(metis
Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
      ultimately have  $\bigwedge C. \text{Prop } C \Psi (!P) (K \langle N \rangle \prec P'')$  using cComm2  $\langle R =$ 
 $!P \rangle \langle \text{guarded } P \rangle$  by simp
    }
  }

```

moreover from $\langle \Psi' \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K \rangle \langle \Psi_P \simeq \mathbf{1} \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$
have $\Psi \vdash M \leftrightarrow K$
by(*metis statEqEnt Identity Composition Commutativity AssertionStatEqSym*)
ultimately have $\text{Prop } C \Psi (P \parallel !P) (\tau \prec (\nu * \text{vec})(P' \parallel P''))$ **using** $\langle \text{vec} \#* \Psi \rangle \langle \text{vec} \#* Q \rangle \langle \text{vec} \#* M \rangle \langle \text{vec} \#* K \rangle \langle \text{vec} \#* C \rangle \langle Q = P \rangle \langle \text{distinct vec} \rangle$
by(*elim rComm2[where K=K and M=M and N=N]*) **auto**
with $\langle R = !P \rangle \langle Q = P \rangle$ **show** *?case by simp*
next
case(*cBrMerge* $\Psi' \Psi_R Q M N P' A_P \Psi_P R P'' A_R \Psi C$)
have $rBrMerge: \bigwedge M N P' P'' C. [\Psi \triangleright P \mapsto \dot{i}M(N) \prec P'; \Psi \triangleright !P \mapsto \dot{i}M(N) \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (\dot{i}M(N) \prec P'')] \implies$
 $\text{Prop } C \Psi (P \parallel !P) (\dot{i}M(N) \prec (P' \parallel P''))$
by fact
from $\langle \text{bangPred } P (Q \parallel R) \rangle$ **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred P (Q || R), auto simp add: psi.inject*)
from $\langle R = !P \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$ **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by**
auto
from $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto \dot{i}M(N) \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$ **have**
 $\Psi \triangleright P \mapsto \dot{i}M(N) \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$ **have**
 $\Psi_P \simeq \mathbf{1}$ **and** $\text{supp } \Psi_P = (\{\} :: \text{name set})$
by(*blast dest: guardedStatEq*)
from $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto \dot{i}M(N) \prec P'' \rangle \langle R = !P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$ **have**
 $\Psi \triangleright !P \mapsto \dot{i}M(N) \prec P''$
by (*metis AssertionStatEqSym Identity compositionSym statEqTransition*)
moreover
{
fix C
have $\text{bangPred } P (!P)$ **by**(*rule aux1*)
moreover from $\langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis*
Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
ultimately have $\bigwedge C. \text{Prop } C \Psi (!P) (\dot{i}M(N) \prec P'')$ **using** *cBrMerge* $\langle R$
 $= !P \rangle \langle \text{guarded } P \rangle$ **by** *simp*
}
ultimately have $\text{Prop } C \Psi (P \parallel !P) (\dot{i}M(N) \prec (P' \parallel P''))$
by(*elim rBrMerge*) **auto**
with $\langle R = !P \rangle \langle Q = P \rangle$ **show** *?case by simp*
next
case(*cBrComm1* $\Psi' \Psi_R Q M N P' A_P \Psi_P R \text{vec } P'' A_R \Psi C$)
have $rBrComm1: \bigwedge M N P' \text{vec } P'' C. [\Psi \triangleright P \mapsto \dot{i}M(N) \prec P'; \Psi \triangleright !P \mapsto \dot{i}M(\nu * \text{vec})(N) \prec P''; \bigwedge C. \text{Prop } C \Psi (!P) (\dot{i}M(\nu * \text{vec})(N) \prec P'');$
 $\text{vec } \#* \Psi; \text{vec } \#* P; \text{vec } \#* M; \text{vec } \#* C;$
 $\text{distinct vec}] \implies \text{Prop } C \Psi (P \parallel !P) (\dot{i}M(\nu * \text{vec})(N) \prec (P' \parallel P''))$
by fact
from $\langle \text{bangPred } P (Q \parallel R) \rangle$ **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred P (Q || R), auto simp add: psi.inject*)
from $\langle R = !P \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$ **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by**

auto
from $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto_i M(N) \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$ **have**
 $\Psi \triangleright P \mapsto_i M(N) \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$ **have**
 $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::\text{name set})$
by(*blast dest: guardedStatEq*)
moreover from $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto_i M(\nu * \text{xvec}) \langle N \rangle \prec P'' \rangle \langle R = !P \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$
 $\langle \Psi \simeq \Psi' \rangle$ **have** $\Psi \triangleright !P \mapsto_i M(\nu * \text{xvec}) \langle N \rangle \prec P''$
by(*metis statEqTransition Identity Composition Commutativity Assertion-StatEqSym*)
moreover
{
fix C
have *bangPred* $P (!P)$ **by**(*rule aux1*)
moreover from $\langle \Psi \simeq \Psi' \rangle \langle \Psi_P \simeq \mathbf{1} \rangle$ **have** $\Psi \simeq \Psi' \otimes \Psi_P$ **by**(*metis Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans*)
ultimately have $\bigwedge C. \text{Prop } C \Psi (!P) (iM(\nu * \text{xvec}) \langle N \rangle \prec P'')$ **using**
cBrComm1 $\langle R = !P \rangle \langle \text{guarded } P \rangle$ **by** *simp*
}
ultimately have *Prop* $C \Psi (P \parallel !P) (iM(\nu * \text{xvec}) \langle N \rangle \prec (P' \parallel P''))$ **using**
 $\langle \text{xvec} \#* \Psi \rangle \langle \text{xvec} \#* Q \rangle \langle \text{xvec} \#* M \rangle \langle \text{xvec} \#* C \rangle \langle Q = P \rangle \langle \text{distinct xvec} \rangle$
by(*elim rBrComm1*) *auto*
with $\langle R = !P \rangle \langle Q = P \rangle$ **show** *?case* **by** *simp*
next
case(*cBrComm2* $\Psi' \Psi_R Q M \text{xvec } N P' A_P \Psi_P R P'' A_R \Psi C$)
have *rBrComm2*: $\bigwedge M N P' \text{xvec } P'' C. [\Psi \triangleright P \mapsto_i M(\nu * \text{xvec}) \langle N \rangle \prec P'; \Psi \triangleright !P \mapsto_i M(N) \prec P'']; \bigwedge C. \text{Prop } C \Psi (!P) (iM(N) \prec P'');$
 $\text{xvec} \#* \Psi; \text{xvec} \#* P; \text{xvec} \#* M; \text{xvec} \#* C;$
 $\text{distinct xvec}] \implies \text{Prop } C \Psi (P \parallel !P) (iM(\nu * \text{xvec}) \langle N \rangle \prec (P' \parallel P''))$
by *fact*
from $\langle \text{bangPred } P (Q \parallel R) \rangle$ **have** $Q = P$ **and** $R = !P$
by $-$ (*ind-cases bangPred P (Q || R), auto simp add: psi.inject*)
from $\langle R = !P \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$ **have** $A_R = []$ **and** $\Psi_R = \mathbf{1}$ **by**
auto
from $\langle \Psi' \otimes \Psi_R \triangleright Q \mapsto_i M(\nu * \text{xvec}) \langle N \rangle \prec P' \rangle \langle Q = P \rangle \langle \Psi \simeq \Psi' \rangle \langle \Psi_R = \mathbf{1} \rangle$
have $\Psi \triangleright P \mapsto_i M(\nu * \text{xvec}) \langle N \rangle \prec P'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover from $\langle Q = P \rangle \langle \text{extractFrame } Q = \langle A_P, \Psi_P \rangle \rangle \langle \text{guarded } P \rangle$ **have**
 $\Psi_P \simeq \mathbf{1}$ **and** *supp* $\Psi_P = (\{\}::\text{name set})$
by(*blast dest: guardedStatEq*)
moreover from $\langle \Psi' \otimes \Psi_P \triangleright R \mapsto_i M(N) \prec P'' \rangle \langle R = !P \rangle \langle \Psi_P \simeq \mathbf{1} \rangle \langle \Psi \simeq \Psi' \rangle$
have $\Psi \triangleright !P \mapsto_i M(N) \prec P''$
by(*metis statEqTransition Identity Composition Commutativity Assertion-StatEqSym*)
moreover
{
fix C
have *bangPred* $P (!P)$ **by**(*rule aux1*)

```

    moreover from ⟨ $\Psi \simeq \Psi'$ ⟩ ⟨ $\Psi_P \simeq \mathbf{1}$ ⟩ have  $\Psi \simeq \Psi' \otimes \Psi_P$  by(metis
    Composition Identity Commutativity AssertionStatEqSym AssertionStatEqTrans)
    ultimately have  $\bigwedge C. Prop C \Psi (!P) (iM(\nu N) \prec P')$  using cBrComm2
    ⟨ $R = !P$ ⟩ ⟨guarded P⟩ by simp
  }
  ultimately have  $Prop C \Psi (P \parallel !P) (iM(\nu *xvec) \langle N \rangle \prec (P' \parallel P'))$  using
  ⟨xvec #*  $\Psi$ ⟩ ⟨xvec #*  $Q$ ⟩ ⟨xvec #*  $M$ ⟩ ⟨xvec #*  $C$ ⟩ ⟨ $Q = P$ ⟩ ⟨distinct xvec⟩
  by(elim rBrComm2) auto
  with ⟨ $R = !P$ ⟩ ⟨ $Q = P$ ⟩ show ?case by simp
next
case(cBrClose  $\Psi' Q M xvec N P' x \Psi C$ )
then show ?case by – (ind-cases bangPred P (( $\nu x$ ) Q))
next
case(cOpen  $\Psi Q M xvec yvec N P' x C$ )
then show ?case by – (ind-cases bangPred P (( $\nu x$ ) Q))
next
case(cBrOpen  $\Psi Q M xvec yvec N P' x C$ )
then show ?case by – (ind-cases bangPred P (( $\nu x$ ) Q))
next
case(cScope  $\Psi Q \alpha P' x C$ )
then show ?case by – (ind-cases bangPred P (( $\nu x$ ) Q))
next
case(Bang  $\Psi' Q Rs \Psi C$ )
have rBang:  $\bigwedge Rs C. [\Psi \triangleright P \parallel !P \mapsto Rs; \bigwedge C. Prop C \Psi (P \parallel !P) Rs;$ 
guarded P]  $\implies Prop C \Psi (!P) Rs$ 
by fact
from ⟨bangPred P (!Q)⟩ have  $P = Q$ 
by – (ind-cases bangPred P (!Q), auto simp add: psi.inject)
with ⟨ $\Psi' \triangleright Q \parallel !Q \mapsto Rs$ ⟩ ⟨ $\Psi \simeq \Psi'$ ⟩ have  $\Psi \triangleright P \parallel !P \mapsto Rs$  by(metis
statEqTransition AssertionStatEqSym)
moreover
{
  fix C
  have bangPred P (P  $\parallel$  !P) by(rule aux2)
  with Bang ⟨ $P = Q$ ⟩ have  $\bigwedge C. Prop C \Psi (P \parallel !P) Rs$  by simp
}
moreover from ⟨guarded Q⟩ ⟨ $P = Q$ ⟩ have guarded P by simp
ultimately have  $Prop C \Psi (!P) Rs$  by(rule rBang)
with ⟨ $P = Q$ ⟩ show ?case by simp
qed
}
with ⟨guarded P⟩ ⟨ $\Psi \triangleright !P \mapsto Rs$ ⟩
show ?thesis by(force intro: aux1)
qed

```

lemma *bangInputInduct*[*consumes 1, case-names cPar1 cPar2 cBang*]:

```

fixes  $\Psi$    :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a

```

and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $Prop$:: 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow 'a \Rightarrow 'a \Rightarrow ('a, 'b, 'c) psi \Rightarrow bool

assumes $\Psi \triangleright !P \mapsto M(N) \prec P'$
and $rPar1$: $\bigwedge P'. \Psi \triangleright P \mapsto M(N) \prec P' \Longrightarrow Prop \Psi (P \parallel !P) M N (P' \parallel !P)$
and $rPar2$: $\bigwedge P'. \llbracket \Psi \triangleright !P \mapsto M(N) \prec P'; Prop \Psi (!P) M N P' \rrbracket \Longrightarrow Prop \Psi (P \parallel !P) M N (P \parallel P')$
and $rBang$: $\bigwedge P'. \llbracket \Psi \triangleright P \parallel !P \mapsto M(N) \prec P'; Prop \Psi (P \parallel !P) M N P'; guarded P \rrbracket \Longrightarrow Prop \Psi (!P) M N P'$
shows $Prop \Psi (!P) M N P'$
using $\langle \Psi \triangleright !P \mapsto M(N) \prec P' \rangle$
by(nominal-induct $\Psi P Rs==M(N) \prec P'$ arbitrary: P' rule: bangInduct)
(auto simp add: residualInject intro: rPar1 rPar2 rBang)

lemma *rbangInputInduct*[consumes 1, case-names cPar1 cPar2 cBrMerge cBang]:
fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $Prop$:: 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow 'a \Rightarrow 'a \Rightarrow ('a, 'b, 'c) psi \Rightarrow bool

assumes $\Psi \triangleright !P \mapsto_i M(N) \prec P'$
and $rPar1$: $\bigwedge P'. \Psi \triangleright P \mapsto_i M(N) \prec P' \Longrightarrow Prop \Psi (P \parallel !P) M N (P' \parallel !P)$
and $rPar2$: $\bigwedge P'. \llbracket \Psi \triangleright !P \mapsto_i M(N) \prec P'; Prop \Psi (!P) M N P' \rrbracket \Longrightarrow Prop \Psi (P \parallel !P) M N (P \parallel P')$
and $rBrMerge$: $\bigwedge P' P'' C. \llbracket \Psi \triangleright P \mapsto_i M(N) \prec P'; \Psi \triangleright !P \mapsto_i M(N) \prec P''; \bigwedge C. Prop \Psi (!P) M N P'' \rrbracket \Longrightarrow Prop \Psi (P \parallel !P) M N (P' \parallel P'')$
and $rBang$: $\bigwedge P'. \llbracket \Psi \triangleright P \parallel !P \mapsto_i M(N) \prec P'; Prop \Psi (P \parallel !P) M N P'; guarded P \rrbracket \Longrightarrow Prop \Psi (!P) M N P'$
shows $Prop \Psi (!P) M N P'$
using $\langle \Psi \triangleright !P \mapsto_i M(N) \prec P' \rangle$
by(nominal-induct $\Psi P Rs==_i M(N) \prec P'$ arbitrary: P' rule: bangInduct)
(auto simp add: residualInject intro: rPar1 rPar2 rBrMerge rBang)

lemma *bangOutputInduct*[consumes 1, case-names cPar1 cPar2 cBang]:
fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and M :: 'a
and $xvec$:: name list
and N :: 'a
and P' :: ('a, 'b, 'c) psi
and $Prop$:: 'd::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow 'a \Rightarrow ('a, 'b, 'c) boundOutput \Rightarrow bool
and C :: 'd

assumes $\Psi \triangleright !P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$
and $rPar1: \bigwedge xvec N P' C. \llbracket \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; distinct\ xvec \rrbracket \implies Prop\ C\ \Psi\ (P \parallel !P)\ M\ ((\nu*xvec)\langle N \rangle \prec' (P' \parallel !P))$
and $rPar2: \bigwedge xvec N P' C. \llbracket \Psi \triangleright !P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; \bigwedge C. Prop\ C\ \Psi\ (!P)\ M\ ((\nu*xvec)\langle N \rangle \prec' P'); xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* C; distinct\ xvec \rrbracket \implies Prop\ C\ \Psi\ (P \parallel !P)\ M\ ((\nu*xvec)\langle N \rangle \prec' (P \parallel P'))$
and $rBang: \bigwedge B C. \llbracket \Psi \triangleright P \parallel !P \mapsto (ROut\ M\ B); \bigwedge C. Prop\ C\ \Psi\ (P \parallel !P)\ M\ B; guarded\ P \rrbracket \implies Prop\ C\ \Psi\ (!P)\ M\ B$
shows $Prop\ C\ \Psi\ (!P)\ M\ ((\nu*xvec)\langle N \rangle \prec' P')$
using $\langle \Psi \triangleright !P \mapsto M(\nu*xvec)\langle N \rangle \prec P' \rangle$
apply(*simp add: residualInject*)
by(*nominal-induct* $\Psi\ P\ Rs==ROut\ M\ ((\nu*xvec)\langle N \rangle \prec' P')$ *avoiding: C arbitrary: xvec N P' rule: bangInduct*)
(force simp add: residualInject intro: rPar1 rPar2 rBang)+

lemma *bangTauInduct*[*consumes 1, case-names cPar1 cPar2 cComm1 cComm2 cBang*]:

fixes $\Psi \quad :: 'b$
and $P \quad :: ('a, 'b, 'c)\ psi$
and $P' \quad :: ('a, 'b, 'c)\ psi$
and $Prop \quad :: 'd::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow bool$
and $C \quad :: 'd$

assumes $\Psi \triangleright !P \mapsto \tau \prec P'$
and $rPar1: \bigwedge P' C. \Psi \triangleright P \mapsto \tau \prec P' \implies Prop\ C\ \Psi\ (P \parallel !P)\ (P' \parallel !P)$
and $rPar2: \bigwedge P' C. \llbracket \Psi \triangleright !P \mapsto \tau \prec P'; \bigwedge C. Prop\ C\ \Psi\ (!P)\ P' \rrbracket \implies Prop\ C\ \Psi\ (P \parallel !P)\ (P \parallel P')$
and $rComm1: \bigwedge M N P' K xvec P'' C. \llbracket \Psi \triangleright P \mapsto M(\langle N \rangle) \prec P'; \Psi \triangleright !P \mapsto K(\nu*xvec)\langle N \rangle \prec P''; \Psi \vdash M \leftrightarrow K; xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* C \rrbracket \implies Prop\ C\ \Psi\ (P \parallel !P)\ ((\nu*xvec)\langle P' \parallel P'' \rangle)$
and $rComm2: \bigwedge M N P' K xvec P'' C. \llbracket \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'; \Psi \triangleright !P \mapsto K(\langle N \rangle) \prec P''; \Psi \vdash M \leftrightarrow K; xvec \#* \Psi; xvec \#* P; xvec \#* M; xvec \#* K; xvec \#* C \rrbracket \implies Prop\ C\ \Psi\ (P \parallel !P)\ ((\nu*xvec)\langle P' \parallel P'' \rangle)$
and $rBang: \bigwedge P' C. \llbracket \Psi \triangleright P \parallel !P \mapsto \tau \prec P'; \bigwedge C. Prop\ C\ \Psi\ (P \parallel !P)\ P'; guarded\ P \rrbracket \implies Prop\ C\ \Psi\ (!P)\ P'$

shows $Prop\ C\ \Psi\ (!P)\ P'$
using $\langle \Psi \triangleright !P \mapsto \tau \prec P' \rangle$
by(*nominal-induct* $\Psi\ P\ Rs==\tau \prec P'$ *avoiding: C arbitrary: P' rule: bangInduct*)
(auto simp add: residualInject intro: rPar1 rPar2 rComm1 rComm2 rBang)

lemma *bangInduct'*[*consumes 2, case-names cAlpha cPar1 cPar2 cComm1 cComm2 cBrMerge cBrComm1 cBrComm2 cBang*]:

fixes $\Psi \quad :: 'b$

and P :: ('a, 'b, 'c) psi
and α :: 'a action
and P' :: ('a, 'b, 'c) psi
and $Prop$:: 'd::fs-name \Rightarrow 'b \Rightarrow ('a, 'b, 'c) psi \Rightarrow 'a action \Rightarrow ('a, 'b, 'c) psi
 \Rightarrow bool
and C :: 'd::fs-name

assumes $\Psi \triangleright !P \mapsto \alpha \prec P'$

and $bn \ \alpha \ \#* \ subject \ \alpha$

and $rAlpha$: $\bigwedge \alpha \ P' \ p \ C. \llbracket bn \ \alpha \ \#* \ \Psi; bn \ \alpha \ \#* \ P; bn \ \alpha \ \#* \ subject \ \alpha; bn \ \alpha \ \#* \ C;$
 $set \ p \subseteq set(bn \ \alpha) \times set(bn(p \cdot \alpha)); distinctPerm \ p;$
 $bn(p \cdot \alpha) \ \#* \ \alpha; bn(p \cdot \alpha) \ \#* \ P'; Prop \ C \ \Psi \ (P \parallel !P) \ \alpha$

$P \rrbracket \Rightarrow$

$Prop \ C \ \Psi \ (P \parallel !P) \ (p \cdot \alpha) \ (p \cdot P')$

and $rPar1$: $\bigwedge \alpha \ P' \ C.$

$\llbracket \Psi \triangleright P \mapsto \alpha \prec P'; bn \ \alpha \ \#* \ \Psi; bn \ \alpha \ \#* \ P; bn \ \alpha \ \#* \ subject \ \alpha; bn \ \alpha \ \#* \ C;$
 $distinct(bn \ \alpha) \rrbracket \Rightarrow$
 $Prop \ C \ \Psi \ (P \parallel !P) \ \alpha \ (P' \parallel !P)$

and $rPar2$: $\bigwedge \alpha \ P' \ C.$

$\llbracket \Psi \triangleright !P \mapsto \alpha \prec P'; \bigwedge C. Prop \ C \ \Psi \ (!P) \ \alpha \ P';$
 $bn \ \alpha \ \#* \ \Psi; bn \ \alpha \ \#* \ P; bn \ \alpha \ \#* \ subject \ \alpha; bn \ \alpha \ \#* \ C; distinct(bn$
 $\alpha) \rrbracket \Rightarrow$

$Prop \ C \ \Psi \ (P \parallel !P) \ \alpha \ (P \parallel P')$

and $rComm1$: $\bigwedge M \ N \ P' \ K \ xvec \ P'' \ C. \llbracket \Psi \triangleright P \mapsto M \langle N \rangle \prec P'; \Psi \triangleright !P$
 $\mapsto K \langle \nu * xvec \rangle \langle N \rangle \prec P''; \bigwedge C. Prop \ C \ \Psi \ (!P) \ (K \langle \nu * xvec \rangle \langle N \rangle) \ P''; \Psi \vdash M \leftrightarrow K;$
 $xvec \ \#* \ \Psi; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ K;$
 $xvec \ \#* \ C; distinct \ xvec \rrbracket \Rightarrow Prop \ C \ \Psi \ (P \parallel !P) \ (\tau) \ ((\nu * xvec) \langle P' \parallel P'' \rangle)$

and $rComm2$: $\bigwedge M \ xvec \ N \ P' \ K \ P'' \ C. \llbracket \Psi \triangleright P \mapsto M \langle \nu * xvec \rangle \langle N \rangle \prec P'; \Psi \triangleright$
 $!P \mapsto K \langle N \rangle \prec P''; \bigwedge C. Prop \ C \ \Psi \ (!P) \ (K \langle N \rangle) \ P''; \Psi \vdash M \leftrightarrow K;$
 $xvec \ \#* \ \Psi; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ K;$
 $xvec \ \#* \ C; distinct \ xvec \rrbracket \Rightarrow Prop \ C \ \Psi \ (P \parallel !P) \ (\tau) \ ((\nu * xvec) \langle P' \parallel P'' \rangle)$

and $rBrMerge$: $\bigwedge M \ N \ P' \ P'' \ C. \llbracket \Psi \triangleright P \mapsto iM \langle N \rangle \prec P'; \Psi \triangleright !P \mapsto iM \langle N \rangle$
 $\prec P''; \bigwedge C. Prop \ C \ \Psi \ (!P) \ (iM \langle N \rangle) \ P'' \rrbracket \Rightarrow$
 $Prop \ C \ \Psi \ (P \parallel !P) \ (iM \langle N \rangle) \ (P' \parallel P'')$

and $rBrComm1$: $\bigwedge M \ N \ P' \ xvec \ P'' \ C. \llbracket \Psi \triangleright P \mapsto iM \langle N \rangle \prec P'; \Psi \triangleright !P$
 $\mapsto iM \langle \nu * xvec \rangle \langle N \rangle \prec P''; \bigwedge C. Prop \ C \ \Psi \ (!P) \ (iM \langle \nu * xvec \rangle \langle N \rangle) \ P'';$
 $xvec \ \#* \ \Psi; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ C;$
 $distinct \ xvec \rrbracket \Rightarrow Prop \ C \ \Psi \ (P \parallel !P) \ (iM \langle \nu * xvec \rangle \langle N \rangle) \ (P' \parallel P'')$

and $rBrComm2$: $\bigwedge M \ N \ P' \ xvec \ P'' \ C. \llbracket \Psi \triangleright P \mapsto iM \langle \nu * xvec \rangle \langle N \rangle \prec P'; \Psi$
 $\triangleright !P \mapsto iM \langle N \rangle \prec P''; \bigwedge C. Prop \ C \ \Psi \ (!P) \ (iM \langle N \rangle) \ P'';$
 $xvec \ \#* \ \Psi; xvec \ \#* \ P; xvec \ \#* \ M; xvec \ \#* \ C;$
 $distinct \ xvec \rrbracket \Rightarrow Prop \ C \ \Psi \ (P \parallel !P) \ (iM \langle \nu * xvec \rangle \langle N \rangle) \ (P' \parallel P'')$

and $rBang$: $\bigwedge \alpha \ P' \ C.$

$\llbracket \Psi \triangleright P \parallel !P \mapsto \alpha \prec P'; guarded \ P; \bigwedge C. Prop \ C \ \Psi \ (P \parallel !P) \ \alpha$
 $P'; guarded \ P; distinct(bn \ \alpha) \rrbracket \Rightarrow$
 $Prop \ C \ \Psi \ (!P) \ \alpha \ P'$

shows $Prop \ C \ \Psi \ (!P) \ \alpha \ P'$

proof –

from $\langle \Psi \triangleright !P \mapsto \alpha \prec P' \rangle$ **have** $distinct(bn \ \alpha)$ **by** ($rule \ boundOutputDistinct$)

with $\langle \Psi \triangleright !P \mapsto \alpha \prec P' \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$ **show** *?thesis*
proof(*nominal-induct* $\Psi P Rs == \alpha \prec P'$ *avoiding: C* $\alpha P'$ *rule: bangInduct*)
case(*cPar1* $\alpha P' C \alpha' P''$)
note $\langle \alpha \prec (P' \parallel !P) = \alpha' \prec P'' \rangle$
moreover from $\langle \text{bn } \alpha \#* \alpha' \rangle$ **have** $\text{bn } \alpha \#* \text{bn } \alpha'$ **by** *simp*
moreover note $\langle \text{distinct}(\text{bn } \alpha) \rangle \langle \text{distinct}(\text{bn } \alpha') \rangle$
moreover from $\langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$ **have** $\text{bn } \alpha \#* (\alpha \prec P' \parallel !P)$ **by** *simp*
moreover from $\langle \text{bn } \alpha' \#* \text{ subject } \alpha' \rangle$ **have** $\text{bn } \alpha' \#* (\alpha' \prec P'')$ **by** *simp*
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha))$ **and**
distinctPerm p **and** $\alpha' = p \cdot \alpha$
and $P'eq: P'' = p \cdot (P' \parallel !P)$ **and** $\text{bn}(p \cdot \alpha) \#* \alpha$ **and** $\text{bn}(p \cdot \alpha) \#* (P' \parallel !P)$
by(*elim residualEq*)

from $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$
have *Prop* $C \Psi (P \parallel !P) \alpha (P' \parallel !P)$
by(*rule rPar1*)

with $\langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle S \langle \text{distinctPerm } p \rangle \langle \text{bn } \alpha \#* \alpha' \rangle \langle \text{bn } \alpha \#* P'' \rangle \langle \alpha' = (p \cdot \alpha) \rangle P'eq \langle \text{bn}(p \cdot \alpha) \#* \alpha \rangle \langle \text{bn}(p \cdot \alpha) \#* (P' \parallel !P) \rangle$
have *Prop* $C \Psi (P \parallel !P) (p \cdot \alpha) (p \cdot (P' \parallel !P))$
by(*elim rAlpha*)
with $P'eq \langle \alpha' = p \cdot \alpha \rangle \langle \text{distinctPerm } p \rangle$ **show** *?case by simp*
next

case(*cPar2* $\alpha P' C \alpha' P''$)
note $\langle \alpha \prec (P \parallel P') = \alpha' \prec P'' \rangle$
moreover from $\langle \text{bn } \alpha \#* \alpha' \rangle$ **have** $\text{bn } \alpha \#* \text{bn } \alpha'$ **by** *simp*
moreover note $\langle \text{distinct}(\text{bn } \alpha) \rangle \langle \text{distinct}(\text{bn } \alpha') \rangle$
moreover from $\langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$ **have** $\text{bn } \alpha \#* (\alpha \prec P \parallel P')$ **by** *simp*
moreover from $\langle \text{bn } \alpha' \#* \text{ subject } \alpha' \rangle$ **have** $\text{bn } \alpha' \#* (\alpha' \prec P'')$ **by** *simp*
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha))$ **and**
distinctPerm p **and** $\alpha' = p \cdot \alpha$
and $P'eq: P'' = p \cdot (P \parallel P')$ **and** $\text{bn}(p \cdot \alpha) \#* \alpha$ **and** $\text{bn}(p \cdot \alpha) \#* (P \parallel P')$
by(*elim residualEq*)

note $\langle \Psi \triangleright !P \mapsto \alpha \prec P' \rangle$
moreover from $\langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$ **have** $\bigwedge C. \text{Prop } C \Psi (!P) \alpha P'$ **by**(*intro cPar2*) *auto*
moreover note $\langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$
ultimately have *Prop* $C \Psi (P \parallel !P) \alpha (P \parallel P')$
by(*rule rPar2*)

with $\langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle \langle \text{bn } \alpha \#* C \rangle S \langle \text{distinctPerm } p \rangle \langle \text{bn } \alpha \#* \alpha' \rangle \langle \text{bn } \alpha \#* P'' \rangle \langle \alpha' = (p \cdot \alpha) \rangle P'eq \langle \text{bn}(p \cdot \alpha) \#* \alpha \rangle \langle \text{bn}(p \cdot \alpha) \#* (P \parallel P') \rangle$
have *Prop* $C \Psi (P \parallel !P) (p \cdot \alpha) (p \cdot (P \parallel P'))$
by(*elim rAlpha*)
with $P'eq \langle \alpha' = p \cdot \alpha \rangle$ **show** *?case by simp*

```

next
  case(cComm1 M N P' K xvec P'' C α P''')
  then have Prop C Ψ (P || !P) (τ) ((ν*xvec)(P' || P''))
    by(elim rComm1) (assumption | simp)+
  then show ?case using ⟨τ < (ν*xvec)(P' || P'') = α < P'''⟩
    by(simp add: residualInject)
next
  case(cComm2 M xvec N P' K P'' C α P''')
  then have Prop C Ψ (P || !P) (τ) ((ν*xvec)(P' || P''))
    by(elim rComm2) (assumption | simp)+
  then show ?case using ⟨τ < (ν*xvec)(P' || P'') = α < P'''⟩
    by(simp add: residualInject)
next
  case(cBrMerge M N P' P'' C α P''')
  then have Prop C Ψ (P || !P) (iM(N)) (P' || P'')
    by(elim rBrMerge) (assumption | simp)+
  then show ?case using ⟨iM(N) < P' || P'' = α < P'''⟩
    by(simp add: residualInject)
next
  case(cBrComm1 M N P' xvec P'' C α P''')
  note ⟨iM(ν*xvec)(N) < (P' || P'') = α < P'''⟩
  moreover from ⟨xvec #* α⟩ have xvec #* bn α by simp
  moreover note ⟨distinct xvec⟩ ⟨distinct(bn α)⟩
  moreover from ⟨xvec #* M⟩ have xvec #* ((iM(ν*xvec)(N)) < (P' || P'')) by
simp
  moreover from ⟨bn α #* subject α⟩ have bn α #* (α < P''') by simp
  ultimately obtain p where S: set p ⊆ set xvec × set(p · xvec) and distinct-
Perm p and α = p · (iM(ν*xvec)(N))
  and P'eq: P''' = p · (P' || P'') and (p · xvec) #* (iM(ν*xvec)(N)) and (p ·
xvec) #* (P' || P'')
  using residualEq[where α=(iM(ν*xvec)(N)) and β=α] by (smt (verit, best)
Sigma-cong bn.simps(4) bnEqvt)

  from cBrComm1 have Prop C Ψ (P || !P) (iM(ν*xvec)(N)) (P' || P'')
    by(elim rBrComm1) (assumption | simp)+

  with ⟨xvec #* Ψ⟩ ⟨xvec #* P⟩ ⟨xvec #* M⟩ ⟨xvec #* C⟩ S ⟨distinctPerm p⟩ ⟨xvec #*
α⟩ ⟨xvec #* P'''⟩ ⟨α = (p · (iM(ν*xvec)(N)))⟩ P'eq ⟨(p · xvec) #* (iM(ν*xvec)(N))⟩
⟨(p · xvec) #* (P' || P'')⟩
  have Prop C Ψ (P || !P) (p · (iM(ν*xvec)(N))) (p · (P' || P''))
    by(intro rAlpha) simp+

  with P'eq ⟨α = p · (iM(ν*xvec)(N))⟩ ⟨distinctPerm p⟩ show ?case by simp
next
  case(cBrComm2 M N P' xvec P'' C α P''')
  note ⟨iM(ν*xvec)(N) < (P' || P'') = α < P'''⟩
  moreover from ⟨xvec #* α⟩ have xvec #* bn α by simp
  moreover note ⟨distinct xvec⟩ ⟨distinct(bn α)⟩
  moreover from ⟨xvec #* M⟩ have xvec #* ((iM(ν*xvec)(N)) < (P' || P'')) by

```

simp

moreover from $\langle \text{bn } \alpha \#* \text{ subject } \alpha \rangle$ **have** $\text{bn } \alpha \#* (\alpha \prec P''')$ **by** *simp*
ultimately obtain p **where** $S: \text{set } p \subseteq \text{set } \text{xvec} \times \text{set}(p \cdot \text{xvec})$ **and** *distinctPerm* p **and** $\alpha = p \cdot (\text{jM}(\nu*\text{xvec})\langle N \rangle)$
and $P'eq: P''' = p \cdot (P' \parallel P'')$ **and** $(p \cdot \text{xvec}) \#* (\text{jM}(\nu*\text{xvec})\langle N \rangle)$ **and** $(p \cdot \text{xvec}) \#* (P' \parallel P'')$
using *residualEq*[**where** $\alpha = (\text{jM}(\nu*\text{xvec})\langle N \rangle)$ **and** $\beta = \alpha$] **by** (*smt* (*verit*, *best*) *Sigma-cong* *bn.simps*(4) *bnEqvt*)

from *cBrComm2* **have** *Prop C Ψ (P || !P) (jM(ν*xvec)⟨N⟩) (P' || P'')*
by(*elim rBrComm2*) (*assumption* | *simp*)+

with $\langle \text{xvec } \#* \Psi \rangle \langle \text{xvec } \#* P \rangle \langle \text{xvec } \#* M \rangle \langle \text{xvec } \#* C \rangle S \langle \text{distinctPerm } p \rangle \langle \text{xvec } \#* \alpha \rangle \langle \text{xvec } \#* P''' \rangle \langle \alpha = (p \cdot (\text{jM}(\nu*\text{xvec})\langle N \rangle)) \rangle P'eq \langle (p \cdot \text{xvec}) \#* (\text{jM}(\nu*\text{xvec})\langle N \rangle) \rangle \langle (p \cdot \text{xvec}) \#* (P' \parallel P'') \rangle$
have *Prop C Ψ (P || !P) (p \cdot (jM(ν*xvec)⟨N⟩)) (p \cdot (P' || P''))*
by(*intro rAlpha*) *simp*+

with $P'eq \langle \alpha = p \cdot (\text{jM}(\nu*\text{xvec})\langle N \rangle) \rangle \langle \text{distinctPerm } p \rangle$ **show** *?case by simp*
next

case(*cBang C α P'*)
then show *?case by(auto intro: rBang)*

qed
qed

lemma *brCommInAuxTooMuch*:

fixes $\Psi :: 'b$
and $\Psi_Q :: 'b$
and $R :: ('a, 'b, 'c) \text{psi}$
and $M :: 'a$
and $N :: 'a$
and $R' :: ('a, 'b, 'c) \text{psi}$
and $A_R :: \text{name list}$
and $\Psi_R :: 'b$
and $A_P :: \text{name list}$
and $\Psi_P :: 'b$
and $A_Q :: \text{name list}$

assumes *RTrans*: $\Psi \otimes \Psi_Q \triangleright R \mapsto \text{jM}(\langle N \rangle) \prec R'$

and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$
and *distinct* A_R
and *QimpP*: $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
and $A_R \#* A_P$
and $A_R \#* A_Q$
and $A_R \#* \Psi$
and $A_R \#* \Psi_P$
and $A_R \#* \Psi_Q$
and $A_R \#* (\Psi \otimes \Psi_Q)$
and $A_R \#* R$

```

and  $A_R \#* M$ 
and  $A_P \#* R$ 
and  $A_P \#* M$ 
and  $A_Q \#* R$ 
and  $A_Q \#* M$ 

shows  $\Psi \otimes \Psi_P \triangleright R \mapsto_{\iota} M(N) \prec R'$ 
  using assms
proof(nominal-induct avoiding: A_P \Psi_P A_Q \Psi_Q \Psi rule: brinputFrameInduct)
  case(cAlpha \Psi' P M N P' A_R \Psi_R p A_P \Psi_P A_Q \Psi_Q \Psi)
  have  $S: \text{set } p \subseteq \text{set } A_R \times \text{set } (p \cdot A_R)$  by fact
  from  $\langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle$ 
  have  $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle)$ 
    by(rule FrameStatImpClosed)
  with  $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle \langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle \langle A_R \#* A_Q \rangle$ 
     $\langle (p \cdot A_R) \#* A_Q \rangle \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S$  distinctPerm p
  have  $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$  by(simp add: eqvts)
  moreover note  $\langle A_R \#* \Psi' \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* M \rangle \langle A_P \#* P \rangle$ 
     $\langle A_Q \#* P \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle$ 
  ultimately show ?case
    by(elim cAlpha)
next
  case(cBrInput \Psi' M K xvec N Tvec P A_P \Psi_P A_Q \Psi_Q \Psi)
  from  $\langle A_P \#* (M(\lambda*xvec N).P) \rangle \langle A_Q \#* (M(\lambda*xvec N).P) \rangle$ 
  have  $A_P \#* M$  and  $A_Q \#* M$  by simp+
  from  $\langle \Psi' \vdash K \succeq M \rangle$ 
  have  $\Psi' \otimes \mathbf{1} \vdash K \succeq M$ 
    by(blast intro: statEqEnt Identity AssertionStatEqSym)
  with  $\langle A_Q \#* K \rangle \langle A_Q \#* M \rangle$ 
  have  $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F K \succeq M$ 
    by(force intro: frameImpI)
  with  $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ 
  have  $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F K \succeq M$ 
    by(simp add: FrameStatImp-def)
  with  $\langle A_P \#* K \rangle \langle A_P \#* M \rangle$  have  $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash K \succeq M$ 
    by(force dest: frameImpE)
  then have  $\Psi \otimes \Psi_P \vdash K \succeq M$  by(blast intro: statEqEnt Identity)
  then show ?case using  $\langle \text{distinct } xvec \rangle \langle \text{set } xvec \subseteq \text{supp } N \rangle \langle \text{length } xvec = \text{length } Tvec \rangle$ 
    by(rule BrInput)
next
  case(cCase \Psi' P M N P' \varphi Cs A_R \Psi_R A_Q \Psi_Q A_P \Psi_P \Psi)
  from  $\langle A_P \#* (\text{Cases } Cs) \rangle \langle A_Q \#* (\text{Cases } Cs) \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$ 
  have  $A_P \#* P$  and  $A_Q \#* P$  and  $A_P \#* \varphi$  and  $A_Q \#* \varphi$ 
    by(auto dest: memFreshChain)

from  $\langle \Psi' \vdash \varphi \rangle$ 

```

have $\Psi' \otimes \mathbf{1} \vdash \varphi$
by(*blast intro: statEqEnt Identity AssertionStatEqSym*)
with $\langle A_P \#* \varphi \rangle$
have $\langle A_P, \Psi' \otimes \mathbf{1} \rangle \vdash_F \varphi$
by(*force intro: frameImpI*)
with $\langle A_P, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1} \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1} \rangle \vdash_F \varphi$
by(*simp add: FrameStatImp-def*)
with $\langle A_Q \#* \varphi \rangle$ **have** $(\Psi \otimes \Psi_Q) \otimes \mathbf{1} \vdash \varphi$
by(*force dest: frameImpE*)
then have $\Psi \otimes \Psi_Q \vdash \varphi$ **by**(*blast intro: statEqEnt Identity*)

have $\langle A_P, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle$
proof –
from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_P, \Psi' \otimes \Psi_R \rangle \simeq_F \langle A_P, \Psi' \otimes \mathbf{1} \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
moreover note $\langle A_P, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1} \rangle$
moreover from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \mathbf{1} \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed

moreover note $\langle A_R \#* \Psi' \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* M \rangle$
 $\langle A_Q \#* P \rangle \langle A_P \#* P \rangle \langle A_Q \#* M \rangle \langle A_P \#* M \rangle \langle A_R \#* A_Q \rangle \langle A_R \#* A_P \rangle$
ultimately have $\Psi \otimes \Psi_Q \triangleright P \mapsto \iota M(N) \prec P'$
by(*elim cCase(4)*)
moreover note $\langle \varphi, P \rangle \in \text{set Cs} \langle \Psi \otimes \Psi_Q \vdash \varphi \rangle \langle \text{guarded } P \rangle$
ultimately show *?case*
by(*rule Case*)

next
case(*cPar1* $\Psi' \Psi_Q P M N P' A_Q Q A_P \Psi_P A_P' \Psi_P' A_Q' \Psi_Q' \Psi$)
from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* A_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle$ **have** $A_P' \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P \rangle$
by (*metis Commutativity FrameStatEq-def frameIntCompositionSym*)
moreover have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P \rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have right: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)
have $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle$

by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNil-StatEq frameImpResChainPres*)
 moreover have $\langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
 ultimately have left: $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis FrameStatImpTrans*)
 from $\langle \langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$ left right
 have $\langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P \rangle$
 by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChain-Pres*)
 moreover note $\langle A_P \#* \Psi' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle$
 $\langle A_P \#* \Psi_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle \langle A_Q' \#* (P \parallel Q) \rangle \langle A_P' \#* \Psi_Q \rangle \langle A_P' \#* M \rangle \langle A_Q' \#* M \rangle \langle A_P \#* A_P' \rangle \langle A_P \#* A_Q' \rangle$
 ultimately have $(\Psi \otimes \Psi_Q) \otimes \Psi_P' \triangleright P \mapsto \imath M(\parallel N) \prec P'$
 by (*elim cPar1(6)*) (*simp | force*)+
 then have $(\Psi \otimes \Psi_P') \otimes \Psi_Q \triangleright P \mapsto \imath M(\parallel N) \prec P'$
 by (*metis associativitySym statEqTransition*)

 moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P' \rangle$
 $\langle A_Q \#* P \rangle \langle A_Q \#* M \rangle \langle A_Q \#* N \rangle$
 ultimately show ?case
 by (*elim Par1*) (*simp | force*)+
 next
 case (*cPar2* $\Psi' \Psi_P Q M N Q' A_P P A_Q \Psi_Q A_P' \Psi_P' A_Q' \Psi_Q' \Psi$)
 from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* A_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle$ have $A_P' \#* \Psi_P$
 by (*force dest: extractFrameFreshChain*)
 have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
 moreover have $\langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P) \otimes \Psi_P') \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def associativitySym frameIntComposition*)
 ultimately have right: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P) \otimes \Psi_P') \otimes \Psi_Q \rangle$
 by (*metis FrameStatImpTrans*)
 moreover have left: $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
 from $\langle \langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$ left right
 have $\langle A_Q', (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P) \otimes \Psi_P') \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
 moreover note $\langle A_Q \#* \Psi' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
 $\langle A_Q \#* \Psi_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle \langle A_Q' \#* (P \parallel Q) \rangle \langle A_P' \#* \Psi_P \rangle \langle A_P' \#* M \rangle \langle A_Q' \#* M \rangle \langle A_Q \#* A_Q' \rangle \langle A_Q \#* A_P' \rangle$
 ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_P' \triangleright Q \mapsto \imath M(\parallel N) \prec Q'$
 by (*elim cPar2(6)*) (*simp | force*)+
 then have $(\Psi \otimes \Psi_P') \otimes \Psi_P \triangleright Q \mapsto \imath M(\parallel N) \prec Q'$

by (*metis associativitySym statEqTransition*)
moreover note $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_P' \rangle$
 $\langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$
ultimately show *?case*
 by(*elim Par2*) (*simp* | *force*)+
next
case(*cBrMerge* $\Psi' \Psi_Q P M N P' A_P \Psi_P Q Q' A_Q A_P' \Psi_P' A_Q' \Psi_Q' \Psi$)
from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* A_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle$ **have** $A_P' \#*$
 Ψ_Q
 by(*force dest: extractFrameFreshChain*)
from $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#* A_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle$ **have** $A_P' \#*$
 Ψ_P
 by(*force dest: extractFrameFreshChain*)
have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P \rangle$
 by (*metis Commutativity FrameStatEq-def frameIntCompositionSym*)
moreover have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q)$
 $\otimes \Psi_P \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P')$
 $\otimes \Psi_P \rangle$
 by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have right: $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q)$
 $\otimes \Psi_P') \otimes \Psi_P \rangle$
 by (*metis FrameStatImpTrans*)
have $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle$
 by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNil-StatEq frameImpResChainPres*)
moreover have $\langle A_Q', \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have left: $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis FrameStatImpTrans*)
from $\langle \langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$ *left right*
have $\langle A_Q', (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_Q) \otimes \Psi_P') \otimes \Psi_P \rangle$
 by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChain-Pres*)
moreover note $\langle A_P \#* \Psi' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P$
 $\#* M \rangle$
 $\langle A_P \#* \Psi_P' \rangle \langle A_P' \#* (P \parallel Q) \rangle \langle A_Q' \#* (P \parallel Q) \rangle \langle A_P' \#* \Psi_Q \rangle \langle A_P' \#* M \rangle \langle A_Q'$
 $\#* M \rangle \langle A_P \#* A_P' \rangle \langle A_P \#* A_Q' \rangle$
ultimately have $(\Psi \otimes \Psi_Q) \otimes \Psi_P' \triangleright P \mapsto \imath M(|N|) \prec P'$
 by(*elim cBrMerge(2)*) (*simp* | *force*)+
then have *Ptrans:* $(\Psi \otimes \Psi_P') \otimes \Psi_Q \triangleright P \mapsto \imath M(|N|) \prec P'$
 by (*metis associativitySym statEqTransition*)

have left2: $\langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_Q', (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)

moreover have $\langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have right2: $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)
from $\langle \langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle$ *left2 right2*
have $\langle A_{Q'}, (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover note $\langle A_Q \#* \Psi' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
 $\langle A_Q \#* \Psi_{P'} \rangle \langle A_{P'} \#* (P \parallel Q) \rangle \langle A_{Q'} \#* (P \parallel Q) \rangle \langle A_{P'} \#* \Psi_P \rangle \langle A_{P'} \#* M \rangle \langle A_{Q'} \#* M \rangle \langle A_Q \#* A_{Q'} \rangle \langle A_Q \#* A_{P'} \rangle$
ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_{P'} \triangleright Q \mapsto \iota M(N) \prec Q'$
by (*elim cBrMerge(6)*) (*simp | force*)
then have $Q_{\text{trans}}: (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \triangleright Q \mapsto \iota M(N) \prec Q'$
by (*metis associativitySym statEqTransition*)

from $P_{\text{trans}} \langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle Q_{\text{trans}} \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_{P'} \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_{P'} \rangle \langle A_Q \#* P \rangle$
 $\langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
show *?case*
by (*elim BrMerge*) (*simp | force*)
next
case (*cScope* $\Psi' P M N P' x A_P \Psi_P A_{P'} \Psi_{P'} A_Q \Psi_Q \Psi$)
then have $\Psi \otimes \Psi_{P'} \triangleright P \mapsto \iota M(N) \prec P'$
by *simp*
with $\langle x \# \Psi \rangle \langle x \# \Psi_{P'} \rangle \langle x \# M \rangle \langle x \# N \rangle$
show *?case*
by (*elim Scope*) (*simp | force*)
next
case (*cBang* $\Psi' P M N P' A_R \Psi_R A_P \Psi_P A_Q \Psi_Q \Psi$)
from $\langle A_R \#* P \rangle$ **have** $A_R \#* (P \parallel !P)$
by *simp*
from $\langle A_P \#* !P \rangle$ **have** $A_P \#* (P \parallel !P)$
by *simp*
from $\langle A_Q \#* !P \rangle$ **have** $A_Q \#* (P \parallel !P)$
by *simp*

have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$
proof –
from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$
by (*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
moreover note $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$
moreover from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$

```

    by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
    ultimately show ?thesis by(rule FrameStatEqImpCompose)
  qed
  then have  $\Psi \otimes \Psi_P \triangleright P \parallel !P \mapsto ;M(N) \prec P'$ 
    using  $\langle A_R \#* \Psi' \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* (P \parallel !P) \rangle \langle A_R \#* \Psi_P \rangle$ 
       $\langle A_P \#* (P \parallel !P) \rangle \langle A_Q \#* (P \parallel !P) \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* M \rangle \langle A_R$ 
 $\#* A_P \rangle \langle A_R \#* A_Q \rangle$ 
    by(elim cBang(5))
  then show ?case using  $\langle \text{guarded } P \rangle$ 
    by(rule Bang)
  qed

```

lemma *brCommInAux*:

```

  fixes  $\Psi$     :: 'b
    and  $\Psi_Q$   :: 'b
    and  $R$     :: ('a, 'b, 'c) psi
    and  $M$     :: 'a
    and  $N$     :: 'a
    and  $R'$    :: ('a, 'b, 'c) psi
    and  $A_R$   :: name list
    and  $\Psi_R$  :: 'b
    and  $A_P$   :: name list
    and  $\Psi_P$  :: 'b
    and  $A_Q$   :: name list

```

assumes *RTrans*: $\Psi \otimes \Psi_Q \triangleright R \mapsto ;M(N) \prec R'$

```

  and FrR: extractFrame  $R = \langle A_R, \Psi_R \rangle$ 
  and distinct  $A_R$ 
  and QimpP:  $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ 
  and  $A_R \#* A_P$ 
  and  $A_R \#* A_Q$ 
  and  $A_R \#* \Psi$ 
  and  $A_R \#* \Psi_P$ 
  and  $A_R \#* \Psi_Q$ 
  and  $A_R \#* R$ 
  and  $A_R \#* M$ 
  and  $A_P \#* R$ 
  and  $A_P \#* M$ 
  and  $A_Q \#* R$ 
  and  $A_Q \#* M$ 

```

shows $\Psi \otimes \Psi_P \triangleright R \mapsto ;M(N) \prec R'$

proof –

```

  from  $\langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_Q \rangle$ 
  have  $A_R \#* (\Psi \otimes \Psi_Q)$ 
    by auto
  with assms
  show ?thesis

```

by(*simp add: brCommInAuxTooMuch*)
qed

lemma *brCommOutAuxTooMuch*:

fixes Ψ :: 'b
 and Ψ_Q :: 'b
 and R :: ('a, 'b, 'c) psi
 and M :: 'a
 and *xvec* :: name list
 and N :: 'a
 and R' :: ('a, 'b, 'c) psi
 and A_R :: name list
 and Ψ_R :: 'b
 and A_P :: name list
 and Ψ_P :: 'b
 and A_Q :: name list

assumes *RTrans*: $\Psi \otimes \Psi_Q \triangleright R \mapsto RBrOut\ M\ (\nu*xvec)\ N\ \prec' R'$
 and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$
 and *distinct* A_R
 and *QimpP*: $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
 and $A_R \#* A_P$
 and $A_R \#* A_Q$
 and $A_R \#* \Psi$
 and $A_R \#* \Psi_P$
 and $A_R \#* \Psi_Q$
 and $A_R \#* (\Psi \otimes \Psi_Q)$
 and $A_R \#* R$
 and $A_R \#* M$
 and $A_P \#* R$
 and $A_P \#* M$
 and $A_Q \#* R$
 and $A_Q \#* M$

shows $\Psi \otimes \Psi_P \triangleright R \mapsto \downarrow M(\nu*xvec)\ \langle N \rangle \prec R'$
 using *assms*

proof(*nominal-induct* $R\ M\ B==(\nu*xvec)\ N\ \prec' R'\ A_R\ \Psi_R$ *avoiding: \Psi\ A_P\ \Psi_P\ A_Q*
 $\Psi_Q\ N\ R'\ xvec$ *rule: broutputFrameInduct*)
 case (*cAlpha* $\Psi' P M A_R \Psi_R p \Psi A_P \Psi_P A_Q \Psi_Q N P' xvec$)
 have S : *set* $p \subseteq \text{set } A_R \times \text{set } (p \cdot A_R)$ **by fact**
 from $\langle \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle \rangle$
 have $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle)$
by(*rule FrameStatImpClosed*)
 with $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle \langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle \langle A_R \#* A_Q \rangle$
 $\langle (p \cdot A_R) \#* A_Q \rangle \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S$ *distinctPerm p*
 have $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ **by**(*simp add: eqvts*)
 moreover **note** $\langle A_R \#* \Psi' \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* M \rangle \langle A_P \#* P \rangle$

```

  ⟨AQ #* P⟩ ⟨AP #* M⟩ ⟨AQ #* M⟩ ⟨AR #* AP⟩ ⟨AR #* AQ⟩
  ultimately show ?case
  by(elim cAlpha)
next
case(cBrOutput Ψ' M K N P Ψ AP ΨP AQ ΨQ N' R' xvec)
from ⟨AP #* (M⟨N⟩.P)⟩ ⟨AQ #* (M⟨N⟩.P)⟩
have AP #* M and AQ #* M by simp+
from ⟨Ψ' ⊢ M ≲ K⟩
have Ψ' ⊗ 1 ⊢ M ≲ K
  by(blast intro: statEqEnt Identity AssertionStatEqSym)
with ⟨AQ #* K⟩ ⟨AQ #* M⟩
have (⟨AQ, Ψ' ⊗ 1⟩) ⊢F M ≲ K
  by(force intro: frameImpI)
with ⟨⟨AQ, Ψ' ⊗ 1⟩ ↦F ⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩⟩
have (⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩) ⊢F M ≲ K
  by(simp add: FrameStatImp-def)
with ⟨AP #* K⟩ ⟨AP #* M⟩ have (Ψ ⊗ ΨP) ⊗ 1 ⊢ M ≲ K
  by(force dest: frameImpE)
then have Ψ ⊗ ΨP ⊢ M ≲ K by(blast intro: statEqEnt Identity)
then have Ψ ⊗ ΨP ▷ M⟨N⟩.P ⟶ ;K⟨N⟩ < P
  by(rule BrOutput)
with ⟨N <' P = (ν*xvec)N' <' R'⟩
show ?case
  by(simp add: residualInject)
next
case(cCase Ψ' R M ϕ Cs AR ΨR Ψ AP ΨP AQ ΨQ N R' xvec)
from ⟨AP #* (Cases Cs)⟩ ⟨AQ #* (Cases Cs)⟩ ⟨(ϕ, R) ∈ set Cs⟩
have AP #* R and AQ #* R and AP #* ϕ and AQ #* ϕ
  by(auto dest: memFreshChain)

from ⟨Ψ' ⊢ ϕ⟩ have Ψ' ⊗ 1 ⊢ ϕ by(blast intro: statEqEnt Identity Assertion-
StatEqSym)
with ⟨AQ #* ϕ⟩ have (⟨AQ, Ψ' ⊗ 1⟩) ⊢F ϕ by(force intro: frameImpI)
with ⟨⟨AQ, Ψ' ⊗ 1⟩ ↦F ⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩⟩ have (⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩)
⊢F ϕ
  by(simp add: FrameStatImp-def)
with ⟨AP #* ϕ⟩ have (Ψ ⊗ ΨP) ⊗ 1 ⊢ ϕ by(force dest: frameImpE)
then have Ψ ⊗ ΨP ⊢ ϕ by(blast intro: statEqEnt Identity)

have ⟨AQ, Ψ' ⊗ ΨR⟩ ↦F ⟨AP, (Ψ ⊗ ΨP) ⊗ ΨR⟩
proof -
  from ⟨ΨR ≲ 1⟩ have ⟨AQ, Ψ' ⊗ ΨR⟩ ≲F ⟨AQ, Ψ' ⊗ 1⟩
  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
  moreover note ⟨⟨AQ, Ψ' ⊗ 1⟩ ↦F ⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩⟩
  moreover from ⟨ΨR ≲ 1⟩ have ⟨AP, (Ψ ⊗ ΨP) ⊗ 1⟩ ≲F ⟨AP, (Ψ ⊗ ΨP) ⊗
ΨR⟩
  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)

```

ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
with $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle$
have $\Psi \otimes \Psi_P \triangleright R \mapsto \downarrow M(\nu*xvec)\langle N \rangle \prec R'$ **using** *cCase*
by(*intro cCase(4)*) *simp+*
then show *?case* **using** $\langle \varphi, R \rangle \in \text{set } Cs \rangle \langle \Psi \otimes \Psi_P \vdash \varphi \rangle \langle \text{guarded } R \rangle$
by(*rule Case*)
next
case(*cPar1* $\Psi' \Psi_Q P M xvec N P' A_Q Q A_P \Psi_P \Psi A_{P'} \Psi_{P'} A_{Q'} \Psi_{Q'} N' R'$
yvec)
from $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* A_{P'} \rangle \langle A_{P'} \#* (P \parallel Q) \rangle$ **have** $A_{P'}$
 $\#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

have $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_Q \otimes \Psi_P \rangle$
by (*metis Commutativity FrameStatEq-def frameIntCompositionSym*)
moreover have $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_Q)$
 $\otimes \Psi_P \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_Q) \otimes \Psi_{P'})$
 $\otimes \Psi_P \rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have right: $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_Q)$
 $\otimes \Psi_{P'}) \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)
have $\langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{Q'}, \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle$
by (*metis AssertionStatEq-def Commutativity compositionSym frameImpNil-StatEq frameImpResChainPres*)
moreover have $\langle A_{Q'}, \Psi' \otimes \Psi_Q \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{Q'}, (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
ultimately have left: $\langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{Q'}, (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle$
by (*metis FrameStatImpTrans*)
from $\langle \langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle$ *left right*
have $\langle A_{Q'}, (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_Q) \otimes \Psi_{P'}) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqSym AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChain-Pres*)
moreover note $\langle A_P \#* \Psi' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P$
 $\#* M \rangle$
 $\langle A_P \#* \Psi_{P'} \rangle \langle A_{P'} \#* (P \parallel Q) \rangle \langle A_{Q'} \#* (P \parallel Q) \rangle \langle A_{P'} \#* \Psi_Q \rangle \langle A_{P'} \#* M \rangle \langle A_{Q'}$
 $\#* M \rangle \langle A_P \#* A_{P'} \rangle \langle A_P \#* A_{Q'} \rangle$
ultimately have $(\Psi \otimes \Psi_Q) \otimes \Psi_{P'} \triangleright P \mapsto \downarrow M(\nu*xvec)\langle N \rangle \prec P'$
by(*intro cPar1(6)*) (*simp | force*)
then have $(\Psi \otimes \Psi_{P'}) \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\nu*xvec)\langle N \rangle \prec P'$
by (*metis associativitySym statEqTransition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle xvec \#* Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#*$
 $\Psi_{P'} \rangle$
 $\langle A_Q \#* P \rangle \langle A_Q \#* M \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* N \rangle$
ultimately have $\Psi \otimes \Psi_{P'} \triangleright P \parallel Q \mapsto \downarrow M(\nu*xvec)\langle N \rangle \prec P' \parallel Q$

by(*elim Par1*) (*simp* | *force*)+
then show *?case using* $\langle (\nu*xvec)N \prec' (P \parallel Q) = (\nu*yvec)N' \prec' R' \rangle$
by(*simp add: residualInject*)
next
case(*cPar2* $\Psi' \Psi_P Q M xvec N Q' A_P P A_Q \Psi_Q \Psi A_{P'} \Psi_{P'} A_{Q'} \Psi_{Q'} N' R' yvec$)
from $\langle extractFrame P = \langle A_P, \Psi_P \rangle \langle A_P \#* A_{P'} \rangle \langle A_{P'} \#* (P \parallel Q) \rangle$ **have** $A_{P'} \#* \Psi_P$
by(*force dest: extractFrameFreshChain*)
have $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
moreover have $\langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def associativitySym frameIntComposition*)
ultimately have right: $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatImpTrans*)
moreover have left: $\langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{Q'}, (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def frameIntAssociativity*)
from $\langle \langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle$ *left right*
have $\langle A_{Q'}, (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_P) \otimes \Psi_{P'}) \otimes \Psi_Q \rangle$
by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
moreover note $\langle A_Q \#* \Psi' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
 $\langle A_Q \#* \Psi_{P'} \rangle \langle A_{P'} \#* (P \parallel Q) \rangle \langle A_{Q'} \#* (P \parallel Q) \rangle \langle A_{P'} \#* \Psi_P \rangle \langle A_{P'} \#* M \rangle \langle A_{Q'} \#* M \rangle \langle A_Q \#* A_{Q'} \rangle \langle A_Q \#* A_{P'} \rangle$
ultimately have $(\Psi \otimes \Psi_P) \otimes \Psi_{P'} \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'$
by(*intro cPar2(6)*) (*simp* | *force*)+
then have $(\Psi \otimes \Psi_{P'}) \otimes \Psi_P \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'$
by (*metis associativitySym statEqTransition*)
moreover note $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle xvec \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_{P'} \rangle$
 $\langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle \langle A_P \#* N \rangle$
ultimately have $\Psi \otimes \Psi_{P'} \triangleright P \parallel Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec P \parallel Q'$
by(*elim Par2*) (*simp* | *force*)+
then show *?case using* $\langle (\nu*xvec)N \prec' (P \parallel Q') = (\nu*yvec)N' \prec' R' \rangle$
by(*simp add: residualInject*)
next
case(*cBrComm1* $\Psi' \Psi_Q P M N P' A_P \Psi_P Q xvec Q' A_Q \Psi A_{P'} \Psi_{P'} A_{Q'} \Psi_{Q'} N' R' yvec$)
have right: $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi_Q \otimes (\Psi \otimes \Psi_{P'})) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity Commutativity FrameStatImpTrans frameImpNilStatEq frameImpResChainPres*)
with $\langle \langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle$
have $\langle A_{Q'}, (\Psi_Q \otimes \Psi') \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{P'}, (\Psi_Q \otimes (\Psi \otimes \Psi_{P'})) \otimes \Psi_P \rangle$
by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity Commutativity FrameStatImpTrans frameImpNilStatEq frameImpResChainPres*)
moreover from $\langle \Psi' \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P' \rangle$
have $\Psi_Q \otimes \Psi' \triangleright P \mapsto_i M(N) \prec P'$

by (*metis Commutativity statEqTransition*)
moreover note $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle \langle \text{distinct } A_P \rangle \langle A_P \#* A_P' \rangle$
 $\langle A_P \#* A_Q' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_P' \rangle \langle A_P \#* \Psi' \rangle \langle A_P \#* P \rangle \langle A_P$
 $\#* M \rangle \langle A_P' \#* (P \parallel Q) \rangle$
 $\langle A_P' \#* M \rangle \langle A_Q' \#* (P \parallel Q) \rangle \langle A_Q' \#* M \rangle$
ultimately have $\Psi_Q \otimes (\Psi \otimes \Psi_P') \triangleright P \mapsto \imath M(\downarrow N) \prec P'$
 by (*elim brCommInAux*) (*simp* | *force*)+
then have $P_{\text{trans}}: (\Psi \otimes \Psi_P') \otimes \Psi_Q \triangleright P \mapsto \imath M(\downarrow N) \prec P'$
 by (*metis Commutativity statEqTransition*)

have $\text{right2}: \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes$
 $\Psi_Q \rangle$
 by (*metis FrameStatEq-def frameIntAssociativity*)
with $\langle \langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_Q', (\Psi' \otimes \Psi_P) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', ((\Psi \otimes \Psi_P') \otimes \Psi_P) \otimes \Psi_Q \rangle$
 by (*metis FrameStatEq-def FrameStatImpTrans frameIntAssociativity*)
then have $Q_{\text{trans}}: (\Psi \otimes \Psi_P') \otimes \Psi_P \triangleright Q \mapsto \imath M(\downarrow \nu * \text{xvec}) \langle N \rangle \prec Q'$
using $\langle A_Q \#* A_P' \rangle \langle A_Q \#* A_Q' \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P' \rangle$
 $\langle A_Q \#* \Psi_P \rangle \langle A_Q \#* \Psi' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle A_P' \#* (P \parallel Q) \rangle$
 $\langle A_P' \#* M \rangle$
 $\langle A_Q' \#* (P \parallel Q) \rangle \langle A_Q' \#* M \rangle$
 by (*intro cBrComm1*) (*simp* | *force*)+
from P_{trans} $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle Q_{\text{trans}}$ $\langle \text{extractFrame } Q = \langle A_Q,$
 $\Psi_Q \rangle \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_P' \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle$
 $\langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P' \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#*$
 $M \rangle \langle \text{xvec} \#* P \rangle$
have $\Psi \otimes \Psi_P' \triangleright P \parallel Q \mapsto \imath M(\downarrow \nu * \text{xvec}) \langle N \rangle \prec P' \parallel Q'$
 by (*elim BrComm1*) (*simp* | *force*)+
then show $?case$ **using** $\langle (\downarrow \nu * \text{xvec}) N \prec' (P' \parallel Q') = (\downarrow \nu * \text{yvec}) N' \prec' R' \rangle$
 by (*simp add: residualInject*)

next
case (*cBrComm2* $\Psi' \Psi_Q P M \text{xvec } N P' A_P \Psi_P Q Q' A_Q \Psi A_P' \Psi_P' A_Q' \Psi_Q' N' R' \text{yvec}$)
have $\langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi_P \otimes (\Psi \otimes \Psi_P')) \otimes \Psi_Q \rangle$
 by (*metis AssertionStatEqTrans AssertionStatEq-def Commutativity associativitySym frameImpNilStatEq frameImpResChainPres*)
with $\langle \langle A_Q', \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi \otimes \Psi_P') \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
have $\langle A_Q', (\Psi_P \otimes \Psi') \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P', (\Psi_P \otimes (\Psi \otimes \Psi_P')) \otimes \Psi_Q \rangle$
 by (*metis AssertionStatEqTrans AssertionStatEq-def Commutativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
moreover from $\langle \Psi' \otimes \Psi_P \triangleright Q \mapsto \imath M(\downarrow N) \prec Q' \rangle$
have $\Psi_P \otimes \Psi' \triangleright Q \mapsto \imath M(\downarrow N) \prec Q'$
 by (*metis Commutativity statEqTransition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle \text{distinct } A_Q \rangle$
 $\langle A_Q \#* A_P' \rangle \langle A_Q \#* A_Q' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P' \rangle$
 $\langle A_Q \#* \Psi' \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle A_P' \#* (P \parallel Q) \rangle \langle A_P' \#* M \rangle \langle A_Q' \#* (P \parallel$
 $Q) \rangle \langle A_Q' \#* M \rangle$
ultimately have $\Psi_P \otimes (\Psi \otimes \Psi_P') \triangleright Q \mapsto \imath M(\downarrow N) \prec Q'$

by (*elim brCommInAux*) (*simp* | *force*)+
 then have $Q_{\text{trans}}: (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \triangleright Q \mapsto \text{!}M(\text{!}N) \prec Q'$
 by (*metis Commutativity statEqTransition*)

have $\langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity associativitySym frameImpNilStatEq frameImpResChainPres*)
 with $\langle \langle A_{Q'}, \Psi' \otimes \Psi_P \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_{P'}, (\Psi \otimes \Psi_{P'}) \otimes \Psi_P \otimes \Psi_Q \rangle \rangle$
 have $\langle A_{Q'}, (\Psi' \otimes \Psi_Q) \otimes \Psi_P \rangle \hookrightarrow_F \langle A_{P'}, ((\Psi \otimes \Psi_{P'}) \otimes \Psi_Q) \otimes \Psi_P \rangle$
 by (*metis AssertionStatEqTrans AssertionStatEq-def Associativity FrameStatImpTrans associativitySym frameImpNilStatEq frameImpResChainPres*)
 with $\langle A_P \#* A_{P'} \rangle \langle A_P \#* A_{Q'} \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_{P'} \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_P \#* \Psi' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_{P'} \#* (P \parallel Q) \rangle \langle A_{P'} \#* M \rangle$
 $\langle A_{Q'} \#* (P \parallel Q) \rangle \langle A_{Q'} \#* M \rangle$
 have $P_{\text{trans}}: (\Psi \otimes \Psi_{P'}) \otimes \Psi_Q \triangleright P \mapsto \text{!}M(\text{!}\nu*xvec)\langle N \rangle \prec P'$
 by (*intro cBrComm2(4)*) (*simp* | *force*)+
 from P_{trans} $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ Q_{trans} $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_{P'} \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle$
 $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_{P'} \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle xvec \#* Q \rangle$
 have $\Psi \otimes \Psi_{P'} \triangleright P \parallel Q \mapsto \text{!}M(\text{!}\nu*xvec)\langle N \rangle \prec P' \parallel Q'$
 by (*elim BrComm2*) (*simp* | *force*)+
 then show $?case$ using $\langle (\text{!}\nu*xvec)\langle N \rangle \prec' (P' \parallel Q') = (\text{!}\nu*yvec)\langle N' \rangle \prec' R' \rangle$
 by (*simp add: residualInject*)

next

case (*cBrOpen* $\Psi' R M' xvec yvec N' R' x A_R \Psi_R \Psi A_P \Psi_P A_Q \Psi_Q N R'' zvec$)
 have $\Psi \otimes \Psi_P \triangleright R \mapsto \text{!}M'(\text{!}\nu*(xvec@yvec))\langle N' \rangle \prec R'$ using *cBrOpen*
 by (*intro cBrOpen(4)*) (*assumption* | *simp*)+

then have $\Psi \otimes \Psi_P \triangleright (\text{!}\nu x)R \mapsto \text{!}M'(\text{!}\nu*(xvec@x\#yvec))\langle N' \rangle \prec R'$
 using $\langle x \in \text{supp } N' \rangle \langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle$
 by (*elim BrOpen*) (*simp* | *force*)+
 with $\langle (\text{!}\nu*(xvec @ x \# yvec))\langle N' \rangle \prec' R' = (\text{!}\nu*zvec)\langle N \rangle \prec' R'' \rangle$
 show $?case$
 by (*simp add: residualInject*)

next

case (*cScope* $\Psi' R M' xvec N' R' x A_R \Psi_R \Psi A_P \Psi_P A_Q \Psi_Q N R'' yvec$)
 then have $\Psi \otimes \Psi_P \triangleright R \mapsto \text{!}M'(\text{!}\nu*xvec)\langle N' \rangle \prec R'$
 by (*intro cScope(4)*) (*simp* | *force*)+
 with $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M' \rangle \langle x \# xvec \rangle \langle x \# N' \rangle$
 have $\Psi \otimes \Psi_P \triangleright (\text{!}\nu x)R \mapsto \text{!}M'(\text{!}\nu*xvec)\langle N' \rangle \prec (\text{!}\nu x)R'$
 by (*elim Scope*) (*simp* | *force*)+
 then show $?case$ using $\langle (\text{!}\nu*xvec)\langle N' \rangle \prec' (\text{!}\nu x)R' = (\text{!}\nu*yvec)\langle N \rangle \prec' R'' \rangle$
 by (*simp add: residualInject*)

next

case (*cBang* $\Psi' R M' A_R \Psi_R \Psi A_P \Psi_P A_Q \Psi_Q N R' xvec$)
 have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$
 proof –
 from $\langle \Psi_R \simeq \mathbf{1} \rangle$ have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$

```

  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
  moreover note  $\langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle$ 
  moreover from  $\langle \Psi_R \simeq \mathbf{1} \rangle$  have  $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$ 
  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
  ultimately show ?thesis by(rule FrameStatEqImpCompose)
qed
then have  $\Psi \otimes \Psi_P \triangleright R \parallel !R \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec R'$  using cBang
  by(intro cBang(5)) (simp | force)
then show ?case using  $\langle \text{guarded } R \rangle$ 
  by(rule Bang)
qed

```

lemma *brCommOutAux*:

```

fixes  $\Psi$    :: 'b
and  $\Psi_Q$   :: 'b
and  $R$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $\text{xvec}$  :: name list
and  $N$      :: 'a
and  $R'$    :: ('a, 'b, 'c) psi
and  $A_R$   :: name list
and  $\Psi_R$  :: 'b
and  $A_P$   :: name list
and  $\Psi_P$  :: 'b
and  $A_Q$   :: name list

```

```

assumes RTrans:  $\Psi \otimes \Psi_Q \triangleright R \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec R'$ 
and FrR: extractFrame  $R = \langle A_R, \Psi_R \rangle$ 
and distinct  $A_R$ 
and QimpP:  $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ 
and  $A_R \#^* A_P$ 
and  $A_R \#^* A_Q$ 
and  $A_R \#^* \Psi$ 
and  $A_R \#^* \Psi_P$ 
and  $A_R \#^* \Psi_Q$ 
and  $A_R \#^* R$ 
and  $A_R \#^* M$ 
and  $A_P \#^* R$ 
and  $A_P \#^* M$ 
and  $A_Q \#^* R$ 
and  $A_Q \#^* M$ 

```

shows $\Psi \otimes \Psi_P \triangleright R \mapsto \text{jM}(\nu * \text{xvec}) \langle N \rangle \prec R'$

proof –

```

  from RTrans have  $\Psi \otimes \Psi_Q \triangleright R \mapsto R\text{BrOut } M (\nu * \text{xvec}) N \prec' R'$ 
  by(simp add: residualInject)

```

moreover from $\langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_Q \rangle$ **have** $A_R \#* (\Psi \otimes \Psi_Q)$
by force
ultimately show *?thesis using assms*
by(*elim brCommOutAuxTooMuch*)
qed

lemma *comm1Aux*:

fixes Ψ $:: 'b$
and Ψ_Q $:: 'b$
and R $:: ('a, 'b, 'c)$ *psi*
and K $:: 'a$
and $xvec$ $::$ *name list*
and N $:: 'a$
and R' $:: ('a, 'b, 'c)$ *psi*
and A_R $::$ *name list*
and Ψ_R $:: 'b$
and P $:: ('a, 'b, 'c)$ *psi*
and M $:: 'a$
and L $:: 'a$
and P' $:: ('a, 'b, 'c)$ *psi*
and A_P $::$ *name list*
and Ψ_P $:: 'b$
and A_Q $::$ *name list*

assumes $RTrans$: $\Psi \otimes \Psi_Q \triangleright R \mapsto K(\nu*xvec)\langle N \rangle \prec R'$
and FrR : $extractFrame R = \langle A_R, \Psi_R \rangle$
and $PTrans$: $\Psi \otimes \Psi_R \triangleright P \mapsto M(L) \prec P'$
and $MeqK$: $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$
and $PeqQ$: $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
and FrP : $extractFrame P = \langle A_P, \Psi_P \rangle$
and FrQ : $extractFrame Q = \langle A_Q, \Psi_Q \rangle$
and *distinct* A_P
and *distinct* A_R
and $A_R \#* A_P$
and $A_R \#* A_Q$
and $A_R \#* \Psi$
and $A_R \#* P$
and $A_R \#* Q$
and $A_R \#* R$
and $A_R \#* K$
and $A_P \#* \Psi$
and $A_P \#* R$
and $A_P \#* P$
and $A_P \#* M$
and $A_Q \#* R$
and $A_Q \#* M$

obtains K' **where** $\Psi \otimes \Psi_P \triangleright R \mapsto K'(\nu*xvec)\langle N \rangle \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash$
 $M \leftrightarrow K'$ **and** $A_R \#* K'$

proof –

from $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle$ *FrP FrQ* **have** $A_R \#* \Psi_P$
and $A_R \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*) +
assume *Assumptions*: $\bigwedge K'. \llbracket \Psi \otimes \Psi_P \triangleright R \mapsto K'(\nu*xvec)\langle N \rangle \prec R'; \Psi \otimes \Psi_P$
 $\otimes \Psi_R \vdash M \leftrightarrow K'; A_R \#* K \rrbracket \implies$ *thesis*
 $\{$
fix $\Psi' :: 'b$
and *zvec :: name list*

assume $A: \Psi \otimes \Psi_Q \simeq \Psi'$
assume $\Psi' \triangleright R \mapsto K(\nu*xvec)\langle N \rangle \prec R'$
then have $\Psi' \triangleright R \mapsto ROut K ((\nu*xvec)\langle N \rangle \prec' R')$ **by**(*simp add: residualInject*)
moreover note *FrR* $\langle distinct A_R \rangle$ *PTrans*
moreover from $\langle \Psi' \triangleright R \mapsto K(\nu*xvec)\langle N \rangle \prec R' \rangle$ **have** *distinct xvec* **by**(*auto*
dest: boundOutputDistinct)
moreover assume $\Psi' \otimes \Psi_R \vdash M \leftrightarrow K$ **and** $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes$
 $\Psi_P) \otimes \Psi_R \rangle$
and $A_R \#* zvec$ **and** $A_P \#* zvec$ **and** $zvec \#* R$ **and** $zvec \#* P$
and $A_R \#* \Psi'$
ultimately have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' ((\nu*xvec)\langle N \rangle \prec' R') \wedge \Psi \otimes$
 $\Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$
using *FrP* $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle distinct A_P \rangle \langle A_R \#*$
 $\Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* K \rangle \langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle \langle A_R \#* \Psi_P \rangle$
proof(*nominal-induct* $\Psi' R K B == (\nu*xvec)\langle N \rangle \prec' R' A_R \Psi_R$ *avoiding: ΨP*
 $A_P \Psi_P A_Q zvec xvec N R'$ *arbitrary: M* *rule: outputFrameInduct*)
case(*cAlpha* $\Psi' R K A_R \Psi_R p \Psi P A_P \Psi_P A_Q zvec xvec N R' M$)
have S : *set* $p \subseteq$ *set* $A_R \times$ *set* $(p \cdot A_R)$ **by fact**
from $\langle \Psi' \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow K \rangle$ **have** $(p \cdot (\Psi' \otimes (p \cdot \Psi_R))) \vdash (p \cdot M) \leftrightarrow$
 $(p \cdot K)$
by(*rule chanEqClosed*)
with $\langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* K \rangle \langle (p \cdot A_R) \#* K \rangle S$ $\langle distinctPerm$
 $p \rangle$
have $\Psi' \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K$ **by**(*simp add: eqvts*)
moreover from $\langle \Psi \otimes (p \cdot \Psi_R) \triangleright P \mapsto M(\downarrow L) \prec P' \rangle S \langle A_R \#* P \rangle \langle (p \cdot A_R)$
 $\#* P \rangle$ **have** $(p \cdot (\Psi \otimes (p \cdot \Psi_R))) \triangleright P \mapsto (p \cdot M)(\downarrow L) \prec P'$
by(*elim inputPermFrameSubject*) *auto*
with $\langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S$ $\langle distinctPerm p \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \mapsto (p$
 $\cdot M)(\downarrow L) \prec P'$
by(*simp add: eqvts*)
moreover from $\langle A_P \#* M \rangle$ **have** $(p \cdot A_P) \#* (p \cdot M)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle S$ **have** $A_P \#* (p \cdot M)$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $(p \cdot A_Q) \#* (p \cdot M)$
by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle A_R \#* A_Q \rangle \langle (p \cdot A_R) \#* A_Q \rangle S$ **have** $A_Q \#* (p \cdot M)$ **by** *simp*
moreover from $\langle \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle \rangle$
have $(p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle)$

by(*rule FrameStatImpClosed*)
with $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle \langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* \Psi_P \rangle$
 $\langle (p \cdot A_R) \#* \Psi_P \rangle \langle A_R \#* A_Q \rangle$
 $\langle (p \cdot A_R) \#* A_Q \rangle \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S \langle \text{distinctPerm } p \rangle$
have $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ **by**(*simp add: eqvts*)
ultimately obtain K' **where** $\Psi \otimes \Psi_P \triangleright R \mapsto \text{ROut } K' ((\nu * \text{xvec})N \prec' R')$
and $\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$ **and** $\text{zvec} \#* K'$ **and** $A_R \#* K'$
using *cAlpha*
by *metis*
from $\langle \Psi \otimes \Psi_P \triangleright R \mapsto \text{ROut } K' ((\nu * \text{xvec})N \prec' R') \rangle S \langle A_R \#* R \rangle \langle (p \cdot A_R) \#* R \rangle$
 $\langle (p \cdot (\Psi \otimes \Psi_P)) \triangleright R \mapsto (\text{ROut } (p \cdot K') ((\nu * \text{xvec})N \prec' R'))$ **using**
outputPermFrameSubject
by(*auto simp add: residualInject*)
with $S \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle$ **have** $\Psi \otimes \Psi_P \triangleright R \mapsto (\text{ROut } (p \cdot K') ((\nu * \text{xvec})N \prec' R'))$
by(*simp add: eqvts*)
moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K' \rangle$ **have** $\langle (p \cdot (\Psi \otimes \Psi_P \otimes \Psi_R)) \vdash (p \cdot p \cdot M) \leftrightarrow (p \cdot K') \rangle$
by(*rule chanEqClosed*)
with $S \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle \langle \text{distinctPerm } p \rangle$ **have** $\Psi \otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow (p \cdot K')$
by(*simp add: eqvts*)
moreover from $\langle \text{zvec} \#* K' \rangle$ **have** $\langle (p \cdot \text{zvec}) \#* (p \cdot K') \rangle$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_R \#* \text{zvec} \rangle \langle (p \cdot A_R) \#* \text{zvec} \rangle S$ **have** $\langle \text{zvec} \#* (p \cdot K') \rangle$ **by** *simp*
moreover from $\langle A_R \#* K' \rangle$ **have** $\langle (p \cdot A_R) \#* (p \cdot K') \rangle$
by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case* **by** *blast*
next
case(*cOutput* $\Psi' M' K N R \Psi P A_P \Psi_P A_Q \text{zvec xvec } N' R' M$)
from $\langle A_P \#* (M' \langle N \rangle . R) \rangle \langle A_Q \#* (M' \langle N \rangle . R) \rangle \langle \text{zvec} \#* (M' \langle N \rangle . R) \rangle$
have $A_P \#* M'$ **and** $A_Q \#* M'$ **and** $\text{zvec} \#* M'$ **by** *simp+*

from $\langle \Psi' \vdash M' \leftrightarrow K \rangle$ **have** $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$ **by**(*blast intro: statEqEnt Identity AssertionStatEqSym*)
then have $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ **by**(*blast intro: chanEqSym chanEqTrans*)
with $\langle A_Q \#* M' \rangle$ **have** $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F M' \leftrightarrow M'$ **by**(*force intro: frameImpI*)

with $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ **have** $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F M' \leftrightarrow M'$
by(*simp add: FrameStatImp-def*)
with $\langle A_P \#* M' \rangle$ **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ **by**(*force dest: frameImpE*)
then have $\Psi \otimes \Psi_P \vdash M' \leftrightarrow M'$ **by**(*blast intro: statEqEnt Identity*) **then**
have $\Psi \otimes \Psi_P \triangleright M' \langle N \rangle . R \mapsto M' \langle N \rangle \prec R$
by(*rule Output*)

moreover from $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow K \rangle \langle \Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K \rangle$
have $\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$ **by**(*metis chanEqSym chanEqTrans*)

with $\langle A_Q \#* M \rangle \langle A_Q \#* M' \rangle$
have $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow M'$
by(*force intro: frameImpI*)
with $\langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle$
have $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow M'$
by(*simp add: FrameStatImp-def*)
with $\langle A_P \#* M \rangle \langle A_P \#* M' \rangle$ **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*force dest: frameImpE*)
then have $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity*)
ultimately show *?case using cOutput* **by**(*auto simp add: residualInject*)
next
case(*cCase* $\Psi' R M' \varphi Cs A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec xvec N R' M$)
from $\langle \text{guarded } R \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$ **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)
with $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M' \rangle$ **have** $\Psi' \otimes \Psi_R \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)
moreover have $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
proof –
from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_Q, \Psi' \otimes \Psi_R \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
moreover note $\langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle$
moreover from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \mathbf{1} \triangleright P \mapsto M(|L|) \prec P' \rangle \langle \Psi_R \simeq \mathbf{1} \rangle$
have $\Psi \otimes \Psi_R \triangleright P \mapsto M(|L|) \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
moreover from $\langle zvec \#* (Cases Cs) \rangle \langle A_P \#* (Cases Cs) \rangle \langle A_Q \#* (Cases Cs) \rangle \langle \varphi, R \rangle \in \text{set } Cs$
have $A_P \#* R$ **and** $A_Q \#* R$ **and** $zvec \#* R$ **and** $A_P \#* \varphi$ **and** $A_Q \#* \varphi$
by(*auto dest: memFreshChain*)
ultimately have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' ((\nu * xvec)N \prec' R') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$ **using** *cCase*
by(*intro cCase*) (*assumption* | *simp*)
then obtain K' **where** $RTrans: \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' ((\nu * xvec)N \prec' R')$
and $MeqK': \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \#* K'$ **and** $A_R \#* K'$
by *metis*
note $RTrans \langle \varphi, R \rangle \in \text{set } Cs$
moreover from $\langle \Psi' \vdash \varphi \rangle$ **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro: statEqEnt Identity AssertionStatEqSym*)
with $\langle A_Q \#* \varphi \rangle$ **have** $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F \varphi$ **by**(*force intro: frameImpI*)
with $\langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle$ **have** $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle)$

$\vdash_F \varphi$
by(*simp add: FrameStatImp-def*)
with $\langle A_P \#* \varphi \rangle$ **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest: frameImpE*)
then have $\Psi \otimes \Psi_P \vdash \varphi$ **by**(*blast intro: statEqEnt Identity*)
ultimately have $\Psi \otimes \Psi_P \triangleright \text{Cases } Cs \mapsto \text{ROut } K' ((\nu*xvec)N \prec' R')$ **using**
 $\langle \text{guarded } R \rangle$ **by**(*rule Case*)
moreover from $\text{Meq}K' \langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*
AssertionStatEqTrans)
ultimately show $?case$ **using** $\langle zvec \#* K' \rangle$
by *auto*
next
case(*cPar1* $\Psi' \Psi_{R2} R_1 M' xvec N' R_1' A_{R2} R_2 A_{R1} \Psi_{R1} \Psi P A_P \Psi_P A_Q$
 $zvec yvec N R' M$)
have *FrR2: extractFrame* $R_2 = \langle A_{R2}, \Psi_{R2} \rangle$ **by** *fact*
from $\langle \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M' \rangle$ **have** $(\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1} \rangle$
proof –
have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans*
AssertionStatEqSym frameNilStatEq frameResChainPres)
moreover note $\langle \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle$
moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans*
AssertionStatEqSym frameNilStatEq frameResChainPres)
ultimately show $?thesis$ **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \triangleright P \mapsto M(\downarrow L) \prec P' \rangle$ **have** $(\Psi \otimes \Psi_{R2}) \otimes \Psi_{R1} \triangleright P \mapsto M(\downarrow L) \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from $\langle A_{R1} \#* A_P \rangle \langle A_{R2} \#* A_P \rangle \langle A_P \#* (R_1 \parallel R_2) \rangle \langle \text{extractFrame } R_1 = \langle A_{R1}, \Psi_{R1} \rangle \rangle$ *FrR2* **have** $A_P \#* \Psi_{R1}$ **and** $A_P \#* \Psi_{R2}$
by(*force dest: extractFrameFreshChain*)
moreover from $\langle (\nu*xvec)N' \prec' (R_1' \parallel R_2) = (\nu*yvec)N \prec' R' \rangle \langle xvec \#* yvec \rangle$
obtain $p T$ **where** $(\nu*xvec)N' \prec' R_1' = (\nu*yvec)N \prec' T$ **and** $R' = T \parallel (p \cdot R_2)$ **and** $set p \subseteq set yvec \times set xvec$
apply(*drule-tac sym*)
by(*elim boundOutputPar1Dest'*) (*assumption* | *simp* | *blast dest: sym*)
ultimately have $\exists K'. (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \triangleright R_1 \mapsto \text{ROut } K' ((\nu*yvec)N \prec' T) \wedge (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K' \wedge (A_{R2} @ zvec) \#* K' \wedge A_{R1} \#* K'$
using *cPar1*
by(*elim cPar1(6)[where ba=P and bb=A_P and bd=A_Q]*) *auto*
then obtain K' **where** *RTrans: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \triangleright R_1 \mapsto K'((\nu*xvec)N \prec' T)$*

$\prec R_1'$
and $MeqK'$: $(\Psi \otimes \Psi_{R_2}) \otimes \Psi_P \otimes \Psi_{R_1} \vdash M \leftrightarrow K'$ **and** $A_{R_2} \#* K'$ **and**
 $A_{R_1} \#* K'$ **and** $zvec \#* K'$
using $\langle (\nu*xvec)N' \prec' R_1' = (\nu*yvec)N \prec' T \rangle$ **by** (*auto simp add: residualInject*)

from $RTrans$ **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R_2} \triangleright R_1 \mapsto K' \langle (\nu*xvec) \langle N' \rangle \prec R_1' \rangle$
by (*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \triangleright (R_1 \parallel R_2) \mapsto K' \langle (\nu*xvec) \langle N' \rangle \prec (R_1' \parallel R_2) \rangle$ **using**
 $FrR2 \langle xvec \#* R_2 \rangle \langle A_{R_2} \#* \Psi \rangle \langle A_{R_2} \#* \Psi_P \rangle \langle A_{R_2} \#* K' \rangle \langle A_{R_2} \#* R_1 \rangle \langle A_{R_2} \#* xvec \rangle \langle A_{R_2} \#* N' \rangle$
by (*force intro: Par1*)
moreover from $MeqK'$ **have** $\Psi \otimes \Psi_P \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M \leftrightarrow K'$
by (*metis statEqEnt Associativity Composition Commutativity*)
ultimately show $?case$ **using** $\langle zvec \#* K' \rangle \langle A_{R_1} \#* K' \rangle \langle A_{R_2} \#* K' \rangle$
 $\langle (\nu*xvec)N' \prec' (R_1' \parallel R_2) = (\nu*yvec)N \prec' R' \rangle$
by (*auto simp add: residualInject*)
next
case ($cPar2 \Psi' \Psi_{R_1} R_2 M' xvec N' R_2' A_{R_1} R_1 A_{R_2} \Psi_{R_2} \Psi_P A_P \Psi_P A_Q$
 $zvec yvec N R' M$)
have $FrR1$: $extractFrame R_1 = \langle A_{R_1}, \Psi_{R_1} \rangle$ **by** *fact*
from $\langle \Psi' \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M \leftrightarrow M' \rangle$ **have** $(\Psi' \otimes \Psi_{R_1}) \otimes \Psi_{R_2} \vdash M \leftrightarrow M'$
by (*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R_1}) \otimes \Psi_{R_2} \rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R_1}) \otimes \Psi_P) \otimes \Psi_{R_2} \rangle$
proof –
have $\langle A_Q, (\Psi' \otimes \Psi_{R_1}) \otimes \Psi_{R_2} \rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rangle$
by (*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
moreover note $\langle A_Q, \Psi' \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rangle$
moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R_1}) \otimes \Psi_P) \otimes \Psi_{R_2} \rangle$
by (*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show $?thesis$ **by** (*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \Psi_{R_1} \otimes \Psi_{R_2} \triangleright P \mapsto M \langle L \rangle \prec P' \rangle$ **have** $(\Psi \otimes \Psi_{R_1}) \otimes \Psi_{R_2} \triangleright P \mapsto M \langle L \rangle \prec P'$
by (*metis statEqTransition Associativity Composition Commutativity*)
moreover from $\langle A_{R_1} \#* A_P \rangle \langle A_{R_2} \#* A_P \rangle \langle A_P \#* (R_1 \parallel R_2) \rangle$ $FrR1 \langle extractFrame R_2 = \langle A_{R_2}, \Psi_{R_2} \rangle \rangle$ **have** $A_P \#* \Psi_{R_1}$ **and** $A_P \#* \Psi_{R_2}$
by (*force dest: extractFrameFreshChain*)
moreover from $\langle (\nu*xvec)N' \prec' (R_1 \parallel R_2') = (\nu*yvec)N \prec' R' \rangle \langle xvec \#* yvec \rangle$
obtain $p T$ **where** $(\nu*xvec)N' \prec' R_2' = (\nu*yvec)N \prec' T$ **and** $R' = (p \cdot R_1) \parallel T$ **and** $set p \subseteq set yvec \times set xvec$
apply (*drule-tac sym*)
by (*elim boundOutputPar2Dest'*) (*assumption | simp | blast dest: sym*)

ultimately have $\exists K'. (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \triangleright R_2 \mapsto ROut K' (\nu*yvec)N \prec' T) \wedge (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K' \wedge (A_{R1}@zvec) \#* K' \wedge A_{R2} \#* K'$
using *cPar2*
by(*elim cPar2(6)*) (*assumption | simp | auto*)+
then obtain K' **where** $RTrans: (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \triangleright R_2 \mapsto K'(\nu*xvec)\langle N' \rangle \prec R_2'$
and $MeqK': (\Psi \otimes \Psi_{R1}) \otimes \Psi_P \otimes \Psi_{R2} \vdash M \leftrightarrow K'$ **and** $A_{R1} \#* K'$ **and** $zvec \#* K'$ **and** $A_{R2} \#* K'$
using $\langle (\nu*xvec)N' \prec' R_2' = (\nu*yvec)N \prec' T \rangle$ **by**(*auto simp add: residualInject*)

from $RTrans$ **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R1} \triangleright R_2 \mapsto K'(\nu*xvec)\langle N' \rangle \prec R_2'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \triangleright (R_1 \parallel R_2) \mapsto K'(\nu*xvec)\langle N' \rangle \prec (R_1 \parallel R_2')$ **using** *FrR1* $\langle xvec \#* R_1 \rangle \langle A_{R1} \#* \Psi \rangle \langle A_{R1} \#* \Psi_P \rangle \langle A_{R1} \#* K' \rangle \langle A_{R1} \#* xvec \rangle \langle A_{R1} \#* N' \rangle \langle A_{R1} \#* R_2 \rangle$
by(*force intro: Par2*)
moreover from $MeqK'$ **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show $?case$ **using** $\langle zvec \#* K' \rangle \langle A_{R1} \#* K' \rangle \langle A_{R2} \#* K' \rangle \langle (\nu*xvec)N' \prec' (R_1 \parallel R_2') = (\nu*yvec)N \prec' R' \rangle$
by(*auto simp add: residualInject*)
next
case(*cOpen* $\Psi' R M' xvec yvec N' R' x A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec zvec2 N R'' M$)
from $\langle (\nu*(xvec @ x \# yvec))N' \prec' R' = (\nu*zvec2)N \prec' R'' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \# zvec2 \rangle \langle x \# R'' \rangle \langle x \# N \rangle \langle distinct zvec2 \rangle$
obtain $xvec' x' yvec'$ **where** $A: (\nu*(xvec@yvec))N' \prec' R' = (\nu*(xvec'@yvec'))(\langle (x, x') \rangle \cdot N) \prec' (\langle (x, x') \rangle \cdot R'')$
and $B: zvec2 = (xvec'@x'\#yvec')$
by(*elim boundOutputOpenDest*) *auto*
then have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' (\nu*(xvec'@yvec'))(\langle (x, x') \rangle \cdot N) \prec' (\langle (x, x') \rangle \cdot R'')$ $\wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge (zvec) \#* K' \wedge A_R \#* K'$ **using** *cOpen*
by(*elim cOpen(4)*) (*assumption | simp*)+
then obtain K' **where** $RTrans: \Psi \otimes \Psi_P \triangleright R \mapsto K'(\nu*(xvec@yvec))\langle N' \rangle \prec R'$
and $MeqK': \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \#* K'$ **and** $A_R \#* K'$
using A **by**(*auto simp add: residualInject*)
from $\langle A_R \#* A_P \rangle \langle A_P \#* (\nu x)R \rangle \langle x \# A_P \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* \Psi_R$
by(*force dest: extractFrameFreshChain*)+
from $\langle \Psi \otimes \Psi_R \triangleright P \mapsto M(L) \prec P' \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle distinct A_P \rangle \langle zvec \#* P \rangle \langle A_P \#* \Psi_R \rangle \langle x \# A_P \rangle \langle A_P \#* M \rangle \langle A_P \#* P \rangle \langle A_P \#* zvec \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* zvec \rangle \langle A_R \#* P \rangle \langle A_R \#* A_P \rangle \langle x \# A_P \rangle \langle x \# P \rangle$
obtain K'' **where** $MeqK'': (\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $A_R \#* K''$ **and** $zvec \#* K''$ **and** $x \# K''$

by(*elim inputObtainPrefix*[**where** $B=(x\#A_R@zvec)$]) (*assumption* | *simp* | *force*)+

from $MeqK'' MeqK'$ **have** $KeqK''$: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$

by(*metis statEqEnt Associativity Composition Commutativity chanEqSym chanEqTrans*)

with $RTrans$ $\langle extractFrame R = \langle A_R, \Psi_R \rangle \langle distinct A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* K' \rangle \langle A_R \#* K'' \rangle \langle A_R \#* R \rangle$

have $\Psi \otimes \Psi_P \triangleright R \mapsto K''(\nu*(xvec@yvec))\langle N' \rangle \prec R'$

by(*elim outputRenameSubject*) (*assumption* | *force*)+

then have $\Psi \otimes \Psi_P \triangleright (\nu x)R \mapsto K''(\nu*(xvec@x\#yvec))\langle N' \rangle \prec R'$

using $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# K'' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle \langle x \in supp N' \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle \langle xvec \#* R \rangle \langle x \# K'' \rangle$

by(*elim Open*) (*assumption* | *force*)+

moreover from $MeqK''$ **have** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$

by(*metis statEqEnt Associativity Composition Commutativity*)

ultimately show $?case$ **using** $\langle zvec \#* K'' \rangle \langle x \# K'' \rangle \langle A_R \#* K'' \rangle B \langle (\nu*(xvec @ x \# yvec))N' \prec' R' = (\nu*zvec2)N \prec' R'' \rangle$

by(*auto simp add: residualInject*)

next

case(*cScope* $\Psi' R M' xvec N' R' x A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec yvec N R'' M$)

from $\langle (\nu*xvec)N' \prec' (\nu x)R' = (\nu*yvec)N \prec' R'' \rangle \langle x \# xvec \rangle \langle x \# yvec \rangle$

obtain R''' **where** $R''' = (\nu x)R'''$ **and** $\langle (\nu*xvec)N' \prec' R' = (\nu*yvec)N \prec' R'' \rangle$

apply(*drule-tac sym*)

by(*metis boundOutputScopeDest*)

then have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' ((\nu*yvec)N \prec' R''') \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$ **using** *cScope*

by(*elim cScope(4)*) (*assumption* | *simp*)+

then obtain K' **where** $RTrans: \Psi \otimes \Psi_P \triangleright R \mapsto K'(\nu*xvec)\langle N' \rangle \prec R'$

and $MeqK': \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \#* K'$ **and** $A_R \#* K'$

using $\langle (\nu*xvec)N' \prec' R' = (\nu*yvec)N \prec' R'' \rangle$

by(*auto simp add: residualInject*)

from $\langle A_R \#* A_P \rangle \langle A_P \#* ((\nu x)R) \rangle \langle x \# A_P \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$

have $A_P \#* \Psi_R$

by(*force dest: extractFrameFreshChain*)+

from $\langle \Psi \otimes \Psi_R \triangleright P \mapsto M(L) \prec P' \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle distinct A_P \rangle \langle x \# P \rangle \langle zvec \#* P \rangle \langle A_P \#* \Psi_R \rangle \langle x \# A_P \rangle \langle A_P \#* M \rangle \langle A_P \#* P \rangle \langle A_P \#* zvec \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* zvec \rangle \langle A_R \#* P \rangle \langle A_R \#* A_P \rangle$

obtain K'' **where** $MeqK'': (\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $x \# K''$ **and** $A_R \#* K''$ **and** $zvec \#* K''$

by(*elim inputObtainPrefix*[**where** $B=(x\#A_R@zvec)$]) (*assumption* | *force*)+

from $MeqK'' MeqK'$ **have** $KeqK''$: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$

by(*metis statEqEnt Associativity Composition Commutativity chanEqSym chanEqTrans*)

with $RTrans$ $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle distinct A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* K' \rangle \langle A_R \#* K'' \rangle \langle A_R \#* R \rangle$

```

have  $\Psi \otimes \Psi_P \triangleright R \mapsto K''(\nu * xvec)\langle N' \rangle \prec R'$ 
  by(elim outputRenameSubject) (assumption | force)+
then have  $\Psi \otimes \Psi_P \triangleright (\nu x)R \mapsto K''(\nu * xvec)\langle N' \rangle \prec (\nu x)R'$  using  $\langle x \# \Psi \rangle$ 
 $\langle x \# \Psi_P \rangle \langle x \# K'' \rangle \langle x \# xvec \rangle \langle x \# N' \rangle \langle xvec \# * \Psi \rangle \langle xvec \# * \Psi_P \rangle \langle xvec \# * R \rangle \langle x \#$ 
 $K'' \rangle$ 
  by(elim Scope) (assumption | force)+
moreover from MeqK'' have  $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$ 
  by(metis statEqEnt Associativity Composition Commutativity)
ultimately show ?case using  $\langle zvec \# * K'' \rangle \langle x \# K'' \rangle \langle A_R \# * K'' \rangle \langle (\nu * xvec)N' \rangle$ 
 $\prec' (\nu x)R' = (\nu * yvec)N \prec' R''$ 
  by(auto simp add: residualInject)
next
case(cBang  $\Psi' R M' A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec xvec N R' M$ )
from  $\langle guarded R \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$  have  $\Psi_R \simeq \mathbf{1}$ 
  by(metis guardedStatEq)
with  $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M' \rangle$  have  $\Psi' \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow M'$ 
  by(metis Identity Commutativity statEqEnt AssertionStatEqSym Composition)
moreover have  $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$ 
proof –
  from  $\langle \Psi_R \simeq \mathbf{1} \rangle$  have  $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$ 
  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
  moreover note  $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ 
  moreover from  $\langle \Psi_R \simeq \mathbf{1} \rangle$  have  $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P)$ 
 $\otimes \Psi_R \otimes \mathbf{1} \rangle$ 
  by(metis Identity Commutativity AssertionStatEqSym Composition frameResChain-
Pres frameNilStatEq AssertionStatEqTrans)
  ultimately show ?thesis by(rule FrameStatEqImpCompose)
qed
moreover from  $\langle \Psi \otimes \mathbf{1} \triangleright P \mapsto M(\downarrow L) \prec P' \rangle \langle \Psi_R \simeq \mathbf{1} \rangle$ 
have  $\Psi \otimes \Psi_R \otimes \mathbf{1} \triangleright P \mapsto M(\downarrow L) \prec P'$  by(metis statEqTransition Identity
Commutativity AssertionStatEqSym Composition)
ultimately have  $\exists K'. \Psi \otimes \Psi_P \triangleright R \parallel !R \mapsto ROut K' ((\nu * xvec)N \prec' R')$ 
 $\wedge \Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K' \wedge zvec \# * K' \wedge A_R \# * K'$  using cBang
  by(intro cBang(5)) (assumption | simp)+
then obtain  $K'$  where RTrans:  $\Psi \otimes \Psi_P \triangleright R \parallel !R \mapsto ROut K' ((\nu * xvec)N$ 
 $\prec' R')$ 
  and MeqK':  $\Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K'$  and  $zvec \# * K'$  and  $A_R \# * K'$ 
  by metis
from RTrans  $\langle guarded R \rangle$  have  $\Psi \otimes \Psi_P \triangleright !R \mapsto ROut K' ((\nu * xvec)N \prec'$ 
 $R')$  by(rule Bang)
moreover from MeqK'  $\langle \Psi_R \simeq \mathbf{1} \rangle$  have  $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$ 
  by(metis Identity Commutativity statEqEnt AssertionStatEqSym Composition
AssertionStatEqTrans)
ultimately show ?case using  $\langle zvec \# * K' \rangle$ 
  by force
qed
}

```

note $Goal = this$
have $\Psi \otimes \Psi_Q \simeq \Psi \otimes \Psi_Q$ **by** *simp*
moreover note $RTrans$
moreover from $MeqK$ **have** $(\Psi \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity*)
moreover note $PeqQ \langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_Q \rangle$
ultimately have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto ROut K' (\nu * xvec) N \prec' R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge ([::name list] \#* K' \wedge A_R \#* K')$
by(*elim Goal*) (*assumption* | *force simp add: residualInject*) +
with *Assumptions* **show** *?thesis*
by(*force simp add: residualInject*)
qed

lemma *comm2Aux*:

fixes $\Psi :: 'b$
and $\Psi_Q :: 'b$
and $R :: ('a, 'b, 'c) psi$
and $K :: 'a$
and $N :: 'a$
and $R' :: ('a, 'b, 'c) psi$
and $A_R :: name list$
and $\Psi_R :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $M :: 'a$
and $xvec :: name list$
and $P' :: ('a, 'b, 'c) psi$
and $A_P :: name list$
and $\Psi_P :: 'b$
and $A_Q :: name list$

assumes $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto K(N) \prec R'$
and $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$
and $PTrans: \Psi \otimes \Psi_R \triangleright P \mapsto M(\nu * xvec) N \prec P'$
and $MeqK: \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$
and $QimpP: \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
and $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$
and $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$
and *distinct* A_P
and *distinct* A_R
and $A_R \#* A_P$
and $A_R \#* A_Q$
and $A_R \#* \Psi$
and $A_R \#* P$
and $A_R \#* Q$
and $A_R \#* R$
and $A_R \#* K$
and $A_P \#* \Psi$
and $A_P \#* R$
and $A_P \#* P$

```

and  $A_P \#* M$ 
and  $A_Q \#* R$ 
and  $A_Q \#* M$ 
and  $A_R \#* xvec$ 
and  $xvec \#* M$ 

obtains  $K'$  where  $\Psi \otimes \Psi_P \triangleright R \mapsto K'(|N|) \prec R'$  and  $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$  and  $A_R \#* K'$ 
proof –
  from  $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle FrP FrQ$  have  $A_R \#* \Psi_P$ 
and  $A_R \#* \Psi_Q$ 
  by(force dest: extractFrameFreshChain)+
  assume Assumptions:  $\bigwedge K'. \llbracket \Psi \otimes \Psi_P \triangleright R \mapsto K'(|N|) \prec R'; \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'; A_R \#* K' \rrbracket \implies thesis$ 
  {
    fix  $\Psi'::'b$ 
    fix zvec::name list
    assume  $A_R \#* \Psi'$ 
    assume  $A_R \#* zvec$ 
    assume  $A_P \#* zvec$ 
    assume  $zvec \#* R$ 
    assume  $zvec \#* P$ 

    assume  $A: \Psi \otimes \Psi_Q \simeq \Psi'$ 
    with RTrans have  $\Psi' \triangleright R \mapsto K(|N|) \prec R'$ 
    by(rule statEqTransition)
    moreover note FrR  $\langle distinct A_R \rangle$ 
    moreover from  $\langle \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$  have  $(\Psi \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$ 
    by(blast intro: statEqEnt Associativity AssertionStatEqSym)
    with  $A$  have  $\Psi' \otimes \Psi_R \vdash M \leftrightarrow K$  by(rule statEqEnt[OF Composition])
    moreover have  $\langle A_Q, \Psi' \otimes \Psi_R \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle$  using  $A$ 
    by(blast dest: frameIntComposition FrameStatEqTrans FrameStatEqSym)
    with QimpP have  $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ 
    by(force intro: FrameStatEqImpCompose)
    moreover from PTrans have distinct xvec by(auto dest: boundOutputDistinct)
    ultimately have  $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto K'(|N|) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$ 
    using PTrans FrP  $\langle A_R \#* K \rangle \langle A_R \#* \Psi' \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$ 
 $\langle A_R \#* \Psi_P \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* \Psi \rangle$ 
 $\langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* zvec \rangle \langle A_Q \#* M \rangle \langle A_R \#* zvec \rangle \langle zvec \#* R \rangle$ 
 $\langle zvec \#* P \rangle \langle distinct A_P \rangle$ 
 $\langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle \langle A_R \#* xvec \rangle \langle xvec \#* M \rangle$ 
    proof(nominal-induct avoiding:  $\Psi P A_P \Psi_P A_Q zvec xvec$  arbitrary: M rule: inputFrameInduct)
    case(cAlpha  $\Psi' R K N R' A_R \Psi_R p \Psi P A_P \Psi_P A_Q zvec xvec M$ )
    have  $S: set p \subseteq set A_R \times set (p \cdot A_R)$  by fact
    from  $\langle \Psi' \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow K \rangle$  have  $(p \cdot (\Psi' \otimes (p \cdot \Psi_R))) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$ 
    by(rule chanEqClosed)
  }

```

with $\langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* K \rangle \langle (p \cdot A_R) \#* K \rangle S \langle \text{distinctPerm } p \rangle$
have $\Psi' \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K$ **by** (*simp add: eqvts*)
moreover from $\langle \Psi \otimes (p \cdot \Psi_R) \triangleright P \mapsto M(\nu * \text{vec}) \langle N \rangle \prec P' \rangle S \langle A_R \#* P \rangle$
 $\langle (p \cdot A_R) \#* P \rangle \langle A_R \#* \text{vec} \rangle \langle (p \cdot A_R) \#* \text{vec} \rangle \langle \text{vec} \#* M \rangle$
have $\langle (p \cdot (\Psi \otimes (p \cdot \Psi_R))) \triangleright P \mapsto (p \cdot M)(\nu * \text{vec}) \langle N \rangle \prec P' \rangle$
using *outputPermFrameSubject* **by** (*auto simp add: residualInject*)
with $\langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S \langle \text{distinctPerm } p \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \mapsto (p \cdot M)(\nu * \text{vec}) \langle N \rangle \prec P'$
by (*simp add: eqvts*)
moreover from $\langle A_P \#* M \rangle$ **have** $\langle (p \cdot A_P) \#* (p \cdot M) \rangle$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle S$ **have** $\langle A_P \#* (p \cdot M) \rangle$ **by** *simp*
moreover from $\langle A_Q \#* M \rangle$ **have** $\langle (p \cdot A_Q) \#* (p \cdot M) \rangle$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_R \#* A_Q \rangle \langle (p \cdot A_R) \#* A_Q \rangle S$ **have** $\langle A_Q \#* (p \cdot M) \rangle$ **by** *simp*

moreover from $\langle \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle \rangle$
have $\langle (p \cdot \langle A_Q, \Psi' \otimes (p \cdot \Psi_R) \rangle) \hookrightarrow_F (p \cdot \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle) \rangle$
by (*rule FrameStatImpClosed*)
with $\langle A_R \#* A_P \rangle \langle (p \cdot A_R) \#* A_P \rangle \langle A_R \#* \Psi' \rangle \langle (p \cdot A_R) \#* \Psi' \rangle \langle A_R \#* \Psi_P \rangle$
 $\langle (p \cdot A_R) \#* \Psi_P \rangle \langle A_R \#* A_Q \rangle$
 $\langle (p \cdot A_R) \#* A_Q \rangle \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S \langle \text{distinctPerm } p \rangle$
have $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ **by** (*simp add: eqvts*)
moreover from $\langle \text{vec} \#* M \rangle$ **have** $\langle (p \cdot \text{vec}) \#* (p \cdot M) \rangle$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $S \langle A_R \#* \text{vec} \rangle \langle (p \cdot A_R) \#* \text{vec} \rangle$ **have** $\langle \text{vec} \#* (p \cdot M) \rangle$ **by** *simp*
ultimately obtain K' **where** $\Psi \otimes \Psi_P \triangleright R \mapsto K' \langle N \rangle \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K'$ **and** $\langle \text{vec} \#* K' \rangle$ **and** $\langle A_R \#* K' \rangle$
using *cAlpha*
by (*auto simp del: freshChainSimps*)
from $\langle \Psi \otimes \Psi_P \triangleright R \mapsto K' \langle N \rangle \prec R' \rangle S \langle A_R \#* R \rangle \langle (p \cdot A_R) \#* R \rangle$ **have** $\langle (p \cdot (\Psi \otimes \Psi_P)) \triangleright R \mapsto (p \cdot K') \langle N \rangle \prec R' \rangle$
by (*elim inputPermFrameSubject*) *auto*
with $S \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle$ **have** $\Psi \otimes \Psi_P \triangleright R \mapsto (p \cdot K') \langle N \rangle \prec R'$
by (*simp add: eqvts*)
moreover from $\langle \Psi \otimes \Psi_P \otimes \Psi_R \vdash (p \cdot M) \leftrightarrow K' \rangle$ **have** $\langle (p \cdot (\Psi \otimes \Psi_P \otimes \Psi_R)) \vdash (p \cdot p \cdot M) \leftrightarrow (p \cdot K') \rangle$
by (*rule chanEqClosed*)
with $S \langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle \langle \text{distinctPerm } p \rangle$
have $\Psi \otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash M \leftrightarrow (p \cdot K')$
by (*simp add: eqvts*)
moreover from $\langle \text{vec} \#* K' \rangle$ **have** $\langle (p \cdot \text{vec}) \#* (p \cdot K') \rangle$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_R \#* \text{vec} \rangle \langle (p \cdot A_R) \#* \text{vec} \rangle S$ **have** $\langle \text{vec} \#* (p \cdot K') \rangle$ **by** *simp*
moreover from $\langle A_R \#* K' \rangle$ **have** $\langle (p \cdot A_R) \#* (p \cdot K') \rangle$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
ultimately show *?case* **by** *blast*

```

next
  case(cInput  $\Psi' M' K \text{vec } N \text{Tvec } R \Psi P A_P \Psi_P A_Q \text{zvec } y\text{vec } M$ )
    from  $\langle A_P \#* (M'(\lambda*\text{vec } N).R) \rangle \langle A_Q \#* (M'(\lambda*\text{vec } N).R) \rangle \langle \text{zvec } \#* (M'(\lambda*\text{vec } N).R) \rangle$ 
    have  $A_P \#* M'$  and  $A_Q \#* M'$  and  $\text{zvec } \#* M'$  by simp+

  from  $\langle \Psi' \vdash M' \leftrightarrow K \rangle$ 
  have  $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K$ 
  by(blast intro: statEqEnt Identity AssertionStatEqSym)
  then have  $\Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ 
  by(blast intro: chanEqSym chanEqTrans)
  with  $\langle A_Q \#* M' \rangle$ 
  have  $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F M' \leftrightarrow M'$ 
  by(force intro: frameImpI)

  with  $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ 
  have  $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F M' \leftrightarrow M'$ 
  by(simp add: FrameStatImp-def)
  with  $\langle A_P \#* M' \rangle$  have  $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M' \leftrightarrow M'$ 
  by(force dest: frameImpE)
  then have  $\Psi \otimes \Psi_P \vdash M' \leftrightarrow M'$  by(blast intro: statEqEnt Identity)
  then have  $\Psi \otimes \Psi_P \triangleright M'(\lambda*\text{vec } N).R \mapsto M'(\lambda*(N[\text{zvec}::=\text{Tvec}])).R \prec R[\text{zvec}::=\text{Tvec}]$ 
  using  $\langle \text{distinct } \text{zvec} \rangle \langle \text{set } \text{zvec} \subseteq \text{supp } N \rangle \langle \text{length } \text{zvec} = \text{length } \text{Tvec} \rangle$ 
  by(rule Input)

  moreover from  $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow K \rangle \langle \Psi' \otimes \mathbf{1} \vdash M' \leftrightarrow K \rangle$ 
  have  $\Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M'$  by(metis chanEqSym chanEqTrans)
  with  $\langle A_Q \#* M \rangle \langle A_Q \#* M' \rangle$ 
  have  $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow M'$ 
  by(force intro: frameImpI)
  with  $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ 
  have  $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F M \leftrightarrow M'$ 
  by(simp add: FrameStatImp-def)
  with  $\langle A_P \#* M \rangle \langle A_P \#* M' \rangle$  have  $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash M \leftrightarrow M'$ 
  by(force dest: frameImpE)
  then have  $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow M'$ 
  by(metis statEqEnt Associativity)
  ultimately show ?case using  $\langle \text{zvec } \#* M' \rangle$ 
  by force
next
  case(cCase  $\Psi' R M' N R' \varphi Cs A_R \Psi_R \Psi P A_P \Psi_P A_Q \text{zvec } \text{zvec } M$ )
  from  $\langle \text{guarded } R \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$  have  $\Psi_R \simeq \mathbf{1}$ 
  by(metis guardedStatEq)
  with  $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M' \rangle$  have  $\Psi' \otimes \Psi_R \vdash M \leftrightarrow M'$ 
  by(metis Identity Commutativity statEqEnt AssertionStatEqSym Composition)
  moreover have  $\langle A_Q, \Psi' \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$ 
  proof -
    from  $\langle \Psi_R \simeq \mathbf{1} \rangle$  have  $\langle A_Q, \Psi' \otimes \Psi_R \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$ 

```

by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
moreover note $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$
moreover from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-Pres frameNilStatEq AssertionStatEqTrans*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \mathbf{1} \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P' \rangle \langle \Psi_R \simeq \mathbf{1} \rangle$
have $\Psi \otimes \Psi_R \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'$ **by**(*metis statEqTransition Identity Commutativity AssertionStatEqSym Composition*)
moreover from $\langle zvec \#* (Cases Cs) \rangle \langle A_P \#* (Cases Cs) \rangle \langle A_Q \#* (Cases Cs) \rangle \langle (\varphi, R) \in set Cs \rangle$
have $A_P \#* R$ **and** $A_Q \#* R$ **and** $zvec \#* R$ **and** $A_P \#* \varphi$ **and** $A_Q \#* \varphi$
by(*auto dest: memFreshChain*)
ultimately have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto K' \langle N \rangle \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$ **using** *cCase*
by(*elim cCase*) (*assumption |simp*)+
then obtain K' **where** $RTrans: \Psi \otimes \Psi_P \triangleright R \mapsto K' \langle N \rangle \prec R'$
and $MeqK': \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \#* K'$ **and** $A_R \#* K'$
by *metis*
note $RTrans \langle (\varphi, R) \in set Cs \rangle$
moreover from $\langle \Psi' \vdash \varphi \rangle$ **have** $\Psi' \otimes \mathbf{1} \vdash \varphi$ **by**(*blast intro: statEqEnt Identity AssertionStatEqSym*)
with $\langle A_Q \#* \varphi \rangle$ **have** $(\langle A_Q, \Psi' \otimes \mathbf{1} \rangle) \vdash_F \varphi$ **by**(*force intro: frameImpI*)
with $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$ **have** $(\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle) \vdash_F \varphi$
by(*simp add: FrameStatImp-def*)
with $\langle A_P \#* \varphi \rangle$ **have** $(\Psi \otimes \Psi_P) \otimes \mathbf{1} \vdash \varphi$ **by**(*force dest: frameImpE*)
then have $\Psi \otimes \Psi_P \vdash \varphi$ **by**(*blast intro: statEqEnt Identity*)
ultimately have $\Psi \otimes \Psi_P \triangleright Cases Cs \mapsto K' \langle N \rangle \prec R'$ **using** $\langle guarded R \rangle$
by(*rule Case*)
moreover from $MeqK' \langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition AssertionStatEqTrans*)
ultimately show *?case* **using** $\langle zvec \#* K' \rangle$
by *force*
next
case(*cPar1* $\Psi' \Psi_{R2} R_1 M' N R_1' A_{R2} R_2 A_{R1} \Psi_{R1} \Psi P A_P \Psi_P A_Q zvec xvec M$)
have $FrR2: extractFrame R_2 = \langle A_{R2}, \Psi_{R2} \rangle$ **by** *fact*
from $\langle \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M' \rangle$ **have** $(\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1} \rangle$
proof –
have $\langle A_Q, (\Psi' \otimes \Psi_{R2}) \otimes \Psi_{R1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans*)

AssertionStatEqSym frameNilStatEq frameResChainPres
moreover note $\langle \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \rangle$
moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R2}) \otimes \Psi_P) \otimes \Psi_{R1} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \Psi_{R1} \otimes \Psi_{R2} \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P' \rangle$ **have** $(\Psi \otimes \Psi_{R2}) \otimes \Psi_{R1} \triangleright P \mapsto M(\nu * xvec) \langle N \rangle \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from $\langle A_{R1} \#* A_P \rangle \langle A_{R2} \#* A_P \rangle \langle A_P \#* (R1 \parallel R2) \rangle \langle extractFrame R1 = \langle A_{R1}, \Psi_{R1} \rangle \rangle$ *FrR2* **have** $A_P \#* \Psi_{R1}$ **and** $A_P \#* \Psi_{R2}$
by(*force dest: extractFrameFreshChain*)
moreover note $\langle distinct\ xvec \rangle$

ultimately have $\exists K'. (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \triangleright R1 \mapsto K' \langle N \rangle \prec R1' \wedge (\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K' \wedge (A_{R2} @ zvec) \#* K' \wedge A_{R1} \#* K'$ **using** *cPar1*
by(*elim cPar1(6)[where ba=P and bb=A_P and bd=A_Q and bf=xvec]*)
auto
then obtain K' **where** *RTrans*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \triangleright R1 \mapsto K' \langle N \rangle \prec R1'$
and *MeqK'*: $(\Psi \otimes \Psi_{R2}) \otimes \Psi_P \otimes \Psi_{R1} \vdash M \leftrightarrow K'$ **and** $A_{R2} \#* K'$ **and** $zvec \#* K'$ **and** $A_{R1} \#* K'$
by *force*

from *RTrans* **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R2} \triangleright R1 \mapsto K' \langle N \rangle \prec R1'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \triangleright (R1 \parallel R2) \mapsto K' \langle N \rangle \prec (R1' \parallel R2)$ **using** *FrR2* $\langle A_{R2} \#* \Psi \rangle \langle A_{R2} \#* \Psi_P \rangle \langle A_{R2} \#* K' \rangle \langle A_{R2} \#* R1 \rangle \langle A_{R2} \#* N \rangle$
by(*force intro: Par1*)
moreover from *MeqK'* **have** $\Psi \otimes \Psi_P \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** $\langle zvec \#* K' \rangle \langle A_{R1} \#* K' \rangle \langle A_{R2} \#* K' \rangle$
by *force*
next
case(*cPar2* $\Psi' \Psi_{R1} R2 M' N R2' A_{R1} R1 A_{R2} \Psi_{R2} \Psi P A_P \Psi_P A_Q zvec$ $xvec\ M$)
have *FrR1*: $extractFrame\ R1 = \langle A_{R1}, \Psi_{R1} \rangle$ **by** *fact*
from $\langle \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \vdash M \leftrightarrow M' \rangle$ **have** $(\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2} \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Composition Commutativity*)
moreover have $\langle A_Q, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2} \rangle \hookrightarrow_F \langle A_P, ((\Psi \otimes \Psi_{R1}) \otimes \Psi_P) \otimes \Psi_{R2} \rangle$
proof –
have $\langle A_Q, (\Psi' \otimes \Psi_{R1}) \otimes \Psi_{R2} \rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
moreover note $\langle \langle A_Q, \Psi' \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R1} \otimes \Psi_{R2} \rangle \rangle$

moreover have $\langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_{R_1} \otimes \Psi_{R_2} \rangle \simeq_F \langle A_P, ((\Psi \otimes \Psi_{R_1}) \otimes \Psi_P) \otimes \Psi_{R_2} \rangle$
by(*metis Associativity Composition Commutativity AssertionStatEqTrans AssertionStatEqSym frameNilStatEq frameResChainPres*)
ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)
qed
moreover from $\langle \Psi \otimes \Psi_{R_1} \otimes \Psi_{R_2} \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P' \rangle$ **have** $(\Psi \otimes \Psi_{R_1}) \otimes \Psi_{R_2} \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P'$
by(*metis statEqTransition Associativity Composition Commutativity*)
moreover from $\langle A_{R_1} \#^* A_P \rangle \langle A_{R_2} \#^* A_P \rangle \langle A_P \#^* (R_1 \parallel R_2) \rangle$ *FrR1* $\langle extractFrame R_2 = \langle A_{R_2}, \Psi_{R_2} \rangle \rangle$ **have** $A_P \#^* \Psi_{R_1}$ **and** $A_P \#^* \Psi_{R_2}$
by(*force dest: extractFrameFreshChain*)
ultimately have $\exists K'. (\Psi \otimes \Psi_{R_1}) \otimes \Psi_P \triangleright R_2 \mapsto K'(\langle N \rangle) \prec R_2' \wedge (\Psi \otimes \Psi_{R_1}) \otimes \Psi_P \otimes \Psi_{R_2} \vdash M \leftrightarrow K' \wedge (A_{R_1} \textcircled{zvec}) \#^* K' \wedge A_{R_2} \#^* K'$ **using** $\langle distinct xvec \rangle$ *cPar2*
by(*elim cPar2(6)[where ba=P and bb=A_P and bd=A_Q and bf=xvec]*)
auto
then obtain K' **where** $RTrans: (\Psi \otimes \Psi_{R_1}) \otimes \Psi_P \triangleright R_2 \mapsto K'(\langle N \rangle) \prec R_2'$
and $MeqK': (\Psi \otimes \Psi_{R_1}) \otimes \Psi_P \otimes \Psi_{R_2} \vdash M \leftrightarrow K'$ **and** $A_{R_1} \#^* K'$ **and** $zvec \#^* K'$ **and** $A_{R_2} \#^* K'$
by force

from $RTrans$ **have** $(\Psi \otimes \Psi_P) \otimes \Psi_{R_1} \triangleright R_2 \mapsto K'(\langle N \rangle) \prec R_2'$
by(*metis statEqTransition Associativity Composition Commutativity*)
then have $\Psi \otimes \Psi_P \triangleright (R_1 \parallel R_2) \mapsto K'(\langle N \rangle) \prec (R_1 \parallel R_2')$ **using** *FrR1* $\langle A_{R_1} \#^* \Psi \rangle \langle A_{R_1} \#^* \Psi_P \rangle \langle A_{R_1} \#^* K' \rangle \langle A_{R_1} \#^* R_2 \rangle \langle A_{R_1} \#^* N \rangle$
by(*force intro: Par2*)
moreover from $MeqK'$ **have** $\Psi \otimes \Psi_P \otimes \Psi_{R_1} \otimes \Psi_{R_2} \vdash M \leftrightarrow K'$
by(*metis statEqEnt Associativity Composition Commutativity*)
ultimately show *?case* **using** $\langle zvec \#^* K' \rangle \langle A_{R_1} \#^* K' \rangle \langle A_{R_2} \#^* K' \rangle$
by force
next
case(*cScope* $\Psi' R M' N R' x A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec xvec M$)
then have $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto K'(\langle N \rangle) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K' \wedge zvec \#^* K' \wedge A_R \#^* K'$
by(*elim cScope(4)*) (*assumption | simp del: freshChainSimps*)
then obtain K' **where** $RTrans: \Psi \otimes \Psi_P \triangleright R \mapsto K'(\langle N \rangle) \prec R'$
and $MeqK': \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $zvec \#^* K'$ **and** $A_R \#^* K'$
by metis
from $\langle A_R \#^* A_P \rangle \langle A_P \#^* ((\nu x)R) \rangle \langle x \#^* A_P \rangle$ $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#^* \Psi_R$
by(*force dest: extractFrameFreshChain*)

from $\langle \Psi \otimes \Psi_R \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P' \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \mapsto ROut M ((\nu^*xvec)\langle N \rangle \prec P')$ **by**(*simp add: residualInject*)
with $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle distinct A_P \rangle \langle x \#^* P \rangle \langle zvec \#^* P \rangle \langle A_P \#^* \Psi_R \rangle \langle x \#^* A_P \rangle \langle A_P \#^* M \rangle \langle A_P \#^* P \rangle \langle A_P \#^* zvec \rangle \langle A_P \#^* \Psi \rangle \langle A_P \#^* zvec \rangle \langle A_R \#^* P \rangle \langle A_R \#^* A_P \rangle \langle xvec \#^* M \rangle \langle distinct xvec \rangle$
obtain K'' **where** $MeqK'': (\Psi \otimes \Psi_R) \otimes \Psi_P \vdash M \leftrightarrow K''$ **and** $x \#^* K''$ **and**

$A_R \#* K''$ and $zvec \#* K''$
by(*elim outputObtainPrefix*[**where** $B=(x\#A_R@zvec)$]) (*assumption* | *simp*
| *force* | *metis freshChainSym*)+

from $MeqK'' MeqK'$ **have** $KeqK''$: $(\Psi \otimes \Psi_P) \otimes \Psi_R \vdash K' \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity chanEqSym*
chanEqTrans)

with $RTrans$ $\langle extractFrame R = \langle A_R, \Psi_R \rangle \langle distinct A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#*$
 $\Psi_P \rangle \langle A_R \#* K' \rangle \langle A_R \#* K'' \rangle \langle A_R \#* R \rangle$
have $\Psi \otimes \Psi_P \triangleright R \mapsto K''(\downarrow N) \prec R'$
by(*elim inputRenameSubject*) (*assumption* | *force*)+

then have $\Psi \otimes \Psi_P \triangleright (\nu x)R \mapsto K''(\downarrow N) \prec (\nu x)R'$ **using** $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$
 $\langle x \# K'' \rangle \langle x \# N \rangle$
by(*elim Scope*) (*assumption* | *force*)+

moreover from $MeqK''$ **have** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K''$
by(*metis statEqEnt Associativity Composition Commutativity*)

ultimately show *?case* **using** $\langle zvec \#* K'' \rangle \langle x \# K'' \rangle \langle A_R \#* K'' \rangle$
by force

next

case(*cBang* $\Psi' R M' N R' A_R \Psi_R \Psi P A_P \Psi_P A_Q zvec xvec M$)
from $\langle guarded R \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$ **have** $\Psi_R \simeq \mathbf{1}$
by(*metis guardedStatEq*)

with $\langle \Psi' \otimes \mathbf{1} \vdash M \leftrightarrow M' \rangle$ **have** $\Psi' \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow M'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*
tion)

moreover have $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \otimes \mathbf{1} \rangle$
proof –

from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_Q, \Psi' \otimes \Psi_R \otimes \mathbf{1} \rangle \simeq_F \langle A_Q, \Psi' \otimes \mathbf{1} \rangle$
by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-*
Pres frameNilStatEq AssertionStatEqTrans)

moreover note $\langle \langle A_Q, \Psi' \otimes \mathbf{1} \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \rangle$
moreover from $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\langle A_P, (\Psi \otimes \Psi_P) \otimes \mathbf{1} \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P)$
 $\otimes \Psi_R \otimes \mathbf{1} \rangle$

by(*metis Identity Commutativity AssertionStatEqSym Composition frameResChain-*
Pres frameNilStatEq AssertionStatEqTrans)

ultimately show *?thesis* **by**(*rule FrameStatEqImpCompose*)

qed

moreover from $\langle \Psi \otimes \mathbf{1} \triangleright P \mapsto M(\nu *xvec)\langle N \rangle \prec P' \rangle \langle \Psi_R \simeq \mathbf{1} \rangle$
have $\Psi \otimes \Psi_R \otimes \mathbf{1} \triangleright P \mapsto M(\nu *xvec)\langle N \rangle \prec P'$ **by**(*metis statEqTransition*
Identity Commutativity AssertionStatEqSym Composition)

ultimately have $\exists K'. \Psi \otimes \Psi_P \triangleright R \parallel !R \mapsto K'(\downarrow N) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R$
 $\otimes \mathbf{1} \vdash M \leftrightarrow K' \wedge zvec \#* K' \wedge A_R \#* K'$ **using** *cBang*

by(*elim cBang(5)*) (*assumption* | *simp del: freshChainSimps*)+

then obtain K' **where** $RTrans$: $\Psi \otimes \Psi_P \triangleright R \parallel !R \mapsto K'(\downarrow N) \prec R'$
and $MeqK'$: $\Psi \otimes \Psi_P \otimes \Psi_R \otimes \mathbf{1} \vdash M \leftrightarrow K'$ **and** $zvec \#* K'$ **and** $A_R \#* K'$
by *metis*

from $RTrans$ $\langle guarded R \rangle$ **have** $\Psi \otimes \Psi_P \triangleright !R \mapsto K'(\downarrow N) \prec R'$ **by**(*rule Bang*)
moreover from $MeqK'$ $\langle \Psi_R \simeq \mathbf{1} \rangle$ **have** $\Psi \otimes \Psi_P \otimes \mathbf{1} \vdash M \leftrightarrow K'$
by(*metis Identity Commutativity statEqEnt AssertionStatEqSym Composition*)

```

AssertionStatEqTrans)
  ultimately show ?case using ⟨zvec #* K'⟩
  by force
qed
}
note Goal = this
have  $\Psi \otimes \Psi_Q \simeq \Psi \otimes \Psi_Q$  by (simp add: AssertionStatEqRefl)
moreover from ⟨ $A_R \#* \Psi$ ⟩ ⟨ $A_R \#* \Psi_Q$ ⟩ have  $A_R \#* (\Psi \otimes \Psi_Q)$  by force
ultimately have  $\exists K'. \Psi \otimes \Psi_P \triangleright R \mapsto K'(|N|) \prec R' \wedge \Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow$ 
 $K' \wedge ([::\text{name list}] \#* K' \wedge A_R \#* K')$ 
  by (intro Goal) (assumption | force)+
with Assumptions show ?thesis
  by blast
qed

```

end

end

```

theory Simulation
  imports Semantics
begin

```

This file is a (heavily modified) variant of the theory *Psi_Calculi.Simulation* from [1].

```

context env begin

```

definition

```

simulation :: 'b  $\Rightarrow$  ('a, 'b, 'c) psi  $\Rightarrow$ 
  ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set  $\Rightarrow$ 
  ('a, 'b, 'c) psi  $\Rightarrow$  bool (-  $\triangleright$  -  $\rightsquigarrow$  [-] - [80, 80, 80, 80] 80)

```

where

```

 $\Psi \triangleright P \rightsquigarrow[\text{Rel}] Q \equiv \forall \alpha Q'. \Psi \triangleright Q \mapsto \alpha \prec Q' \longrightarrow \text{bn } \alpha \#* \Psi \longrightarrow \text{bn } \alpha \#* P$ 
 $\longrightarrow (\exists P'. \Psi \triangleright P \mapsto \alpha \prec P' \wedge (\Psi, P', Q') \in \text{Rel})$ 

```

abbreviation

```

simulationNilJudge (-  $\rightsquigarrow$  [-] - [80, 80, 80] 80) where  $P \rightsquigarrow[\text{Rel}] Q \equiv \text{SBottom}'$ 
 $\triangleright P \rightsquigarrow[\text{Rel}] Q$ 

```

lemma simI[consumes 1, case-names cSim]:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\text{Rel}$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$  :: ('a, 'b, 'c) psi
and  $C$  :: 'd::fs-name

```

assumes Eqvt: eqvt Rel

```

and Sim:  $\bigwedge \alpha Q'. [\Psi \triangleright Q \mapsto \alpha \prec Q'; \text{bn } \alpha \#* P; \text{bn } \alpha \#* Q; \text{bn } \alpha \#* \Psi;$ 
distinct(bn  $\alpha$ );

```

```

 $\text{bn } \alpha \#* (\text{subject } \alpha); \text{bn } \alpha \#* C] \Longrightarrow \exists P'. \Psi \triangleright P \mapsto \alpha \prec P'$ 

```

$\wedge (\Psi, P', Q') \in Rel$

shows $\Psi \triangleright P \rightsquigarrow[Rel] Q$

proof –

```

{
  fix  $\alpha Q'$ 
  assume  $\Psi \triangleright Q \mapsto \alpha \prec Q'$  and  $bn \alpha \#* \Psi$  and  $bn \alpha \#* P$ 
  then have  $\exists P'. \Psi \triangleright P \mapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$ 
  proof (nominal-induct  $\alpha$  rule: action.strong-induct)
    case (In M N)
    then show ?case by (auto simp add: Sim)
  next
    case (BrIn M N)
    then show ?case by (auto simp add: Sim)
  next
    case (Out M xvec N)
  moreover {
    fix M xvec N Q'
    assume (xvec::name list)  $\#* \Psi$  and xvec  $\#* P$ 
    obtain p where xvecFreshPsi: ((p::name prm) · (xvec::name list))  $\#* \Psi$ 
    and xvecFreshM: (p · xvec)  $\#* (M::'a)$ 
    and xvecFreshN: (p · xvec)  $\#* (N::'a)$ 
    and xvecFreshP: (p · xvec)  $\#* P$ 
    and xvecFreshQ: (p · xvec)  $\#* Q$ 
    and xvecFreshQ': (p · xvec)  $\#* (Q'::('a, 'b, 'c) psi)$ 
    and xvecFreshC: (p · xvec)  $\#* C$ 
    and xvecFreshxvec: (p · xvec)  $\#* xvec$ 
    and S: (set p)  $\subseteq$  (set xvec)  $\times$  (set (p · xvec))
    and dpr: distinctPerm p
    by (rule name-list-avoiding[where xvec=xvec and c=( $\Psi, M, Q, N, P, Q',$ 
xvec, C)]) auto
    from  $\langle (p \cdot xvec) \#* M \rangle \langle distinctPerm p \rangle$  have xvec  $\#* (p \cdot M)$ 
    by (subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, where pi=p,
symmetric]) simp

    assume Trans:  $\Psi \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec Q'$ 
    with xvecFreshN xvecFreshQ' S
    have  $\Psi \triangleright Q \mapsto M(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot Q')$ 
    by (simp add: boundOutputChainAlpha'' residualInject)
    moreover then have distinct(p · xvec) by (auto dest: boundOutputDistinct)

  moreover note xvecFreshPsi xvecFreshP xvecFreshQ xvecFreshM xvecFreshC
  ultimately obtain P' where PTrans:  $\Psi \triangleright P \mapsto M(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle$ 
 $\prec P'$ 
  and P'RelQ':  $(\Psi, P', p \cdot Q') \in Rel$ 
  using Sim by (metis bn.simps(3) optionFreshChain(1) subject.simps(3))
  then have (p ·  $\Psi$ )  $\triangleright$  (p · P)  $\mapsto$  (p · (M( $\nu * (p \cdot xvec)$ ))  $\langle (p \cdot N) \rangle \prec P')$ 
  by (simp add: semantics.eqvt)
  with  $\langle xvec \#* \Psi \rangle$  xvecFreshPsi  $\langle xvec \#* P \rangle$  xvecFreshP S dpr

```

```

have  $\Psi \triangleright P \mapsto (p \cdot M) \langle \nu * xvec \rangle \langle N \rangle \prec (p \cdot P')$ 
  by (simp add: eqvts name-set-fresh-fresh)
with  $\langle xvec \#* \Psi \rangle xvecFreshPsi \langle xvec \#* P \rangle xvecFreshP S \langle xvec \#* (p \cdot M) \rangle$ 
have  $\Psi \triangleright P \mapsto (p \cdot p \cdot M) \langle \nu * xvec \rangle \langle N \rangle \prec (p \cdot P')$ 
  by (simp add: outputPermSubject)

with dpr have  $\Psi \triangleright P \mapsto M \langle \nu * xvec \rangle \langle N \rangle \prec (p \cdot P')$ 
  by simp

moreover from  $P' Rel Q' Eqvt$  have  $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$ 
  apply (simp add: eqvt-def eqvts)
  apply (erule ballE[where x=( $\Psi, P', p \cdot Q'$ )])
  apply (erule allE[where x=p])
  by (auto simp add: eqvts)

with  $\langle xvec \#* \Psi \rangle xvecFreshPsi S$  dpr have  $(\Psi, p \cdot P', Q') \in Rel$ 
  by simp
ultimately have  $\exists P'. \Psi \triangleright P \mapsto M \langle \nu * xvec \rangle \langle N \rangle \prec P' \wedge (\Psi, P', Q') \in$ 
Rel
  by blast
}
ultimately show ?case by force
next
case (BrOut  $M xvec N$ )
moreover {
  fix  $M xvec N Q'$ 
  assume  $(xvec::name\ list) \#* \Psi$  and  $xvec \#* P$ 
  obtain  $p$  where  $xvecFreshPsi: ((p::name\ prm) \cdot (xvec::name\ list)) \#* \Psi$ 
  and  $xvecFreshM: (p \cdot xvec) \#* (M::'a)$ 
  and  $xvecFreshN: (p \cdot xvec) \#* (N::'a)$ 
  and  $xvecFreshP: (p \cdot xvec) \#* P$ 
  and  $xvecFreshQ: (p \cdot xvec) \#* Q$ 
  and  $xvecFreshQ': (p \cdot xvec) \#* (Q'::('a, 'b, 'c)\ psi)$ 
  and  $xvecFreshC: (p \cdot xvec) \#* C$ 
  and  $xvecFreshxvec: (p \cdot xvec) \#* xvec$ 
  and  $S: (set\ p) \subseteq (set\ xvec) \times (set\ (p \cdot xvec))$ 
  and  $dpr: distinctPerm\ p$ 
  by (rule name-list-avoiding[where xvec=xvec and c=( $\Psi, M, Q, N, P, Q',$ 
xvec, C)]) auto

  from  $\langle (p \cdot xvec) \#* M \rangle \langle distinctPerm\ p \rangle$  have  $xvec \#* (p \cdot M)$ 
  by (subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, where pi=p,
symmetric]) simp

  assume  $Trans: \Psi \triangleright Q \mapsto_i M \langle \nu * xvec \rangle \langle N \rangle \prec Q'$ 
  with  $xvecFreshN xvecFreshQ' S$ 
  have  $\Psi \triangleright Q \mapsto_i M \langle \nu * (p \cdot xvec) \rangle \langle (p \cdot N) \rangle \prec (p \cdot Q')$ 
  by (simp add: boundOutputChainAlpha'' residualInject)
  moreover then have  $distinct(p \cdot xvec)$ 

```

```

    by(auto dest: boundOutputDistinct)

    moreover note xvecFreshPsi xvecFreshP xvecFreshQ xvecFreshM xvecFreshC
    ultimately obtain P' where PTrans:  $\Psi \triangleright P \mapsto_{iM} (\nu^*(p \cdot xvec)) \langle (p \cdot N) \rangle \prec P'$ 
    and P'RelQ':  $(\Psi, P', p \cdot Q') \in Rel$ 
    by(auto dest: Sim)
    then have  $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot (iM(\nu^*(p \cdot xvec)) \langle (p \cdot N) \rangle \prec P'))$ 
    by(metis semantics.eqvt)
    with  $\langle xvec \ \#* \ \Psi \rangle$  xvecFreshPsi  $\langle xvec \ \#* \ P \rangle$  xvecFreshP S dpr
    have  $\Psi \triangleright P \mapsto_{iM} (p \cdot M) (\nu^* xvec) \langle N \rangle \prec (p \cdot P')$ 
    by(simp add: eqvts name-set-fresh-fresh)
    with  $\langle xvec \ \#* \ \Psi \rangle$  xvecFreshPsi  $\langle xvec \ \#* \ P \rangle$  xvecFreshP S  $\langle xvec \ \#* \ (p \cdot M) \rangle$ 
    have  $\Psi \triangleright P \mapsto_{iM} (p \cdot p \cdot M) (\nu^* xvec) \langle N \rangle \prec (p \cdot P')$ 
    by(simp add: broutputPermSubject)

    with dpr have  $\Psi \triangleright P \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec (p \cdot P')$ 
    by simp

    moreover from P'RelQ' Eqvt have  $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$ 
    apply(simp add: eqvt-def eqvts)
    apply(erule ballE[where x=( $\Psi, P', p \cdot Q'$ )])
    apply(erule allE[where x=p])
    by(auto simp add: eqvts)

    with  $\langle xvec \ \#* \ \Psi \rangle$  xvecFreshPsi S dpr have  $(\Psi, p \cdot P', Q') \in Rel$ 
    by simp
    ultimately have  $\exists P'. \Psi \triangleright P \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec P' \wedge (\Psi, P', Q') \in Rel$ 
    by blast
  }
  ultimately show ?case by force
next
case Tau
  then show ?case by (auto simp add: Sim)
qed
}
then show ?thesis
  by (auto simp add: simulation-def)
qed

lemma simI2[case-names cSim]:
  fixes  $\Psi$  :: 'b
  and P :: ('a, 'b, 'c) psi
  and Rel :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
  and Q :: ('a, 'b, 'c) psi
  and C :: 'd::fs-name

  assumes Sim:  $\bigwedge \alpha Q'. [\Psi \triangleright Q \mapsto_{\alpha} \prec Q'; bn \ \alpha \ \#* \ P; bn \ \alpha \ \#* \ \Psi; distinct(bn \ \alpha)]$ 

```

$\implies \exists P'. \Psi \triangleright P \mapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

shows $\Psi \triangleright P \rightsquigarrow [Rel] Q$

using *assms*

by(*auto simp add: simulation-def dest: boundOutputDistinct*)

lemma *simIChainFresh*[*consumes 4, case-names cSim*]:

fixes Ψ :: 'b

and P :: ('a, 'b, 'c) psi

and Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set

and Q :: ('a, 'b, 'c) psi

and $xvec$:: name list

and C :: 'd::fs-name

assumes *Eqt*: *eqvt Rel*

and $xvec \#* \Psi$

and $xvec \#* P$

and $xvec \#* Q$

and Sim : $\bigwedge \alpha Q'. \llbracket \Psi \triangleright Q \mapsto \alpha \prec Q'; bn \alpha \#* P; bn \alpha \#* Q; bn \alpha \#* \Psi;$
 $bn \alpha \#* subject \alpha; distinct(bn \alpha); bn \alpha \#* C; xvec \#* \alpha; xvec$

$\#* Q \rrbracket \implies$

$\exists P'. \Psi \triangleright P \mapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

shows $\Psi \triangleright P \rightsquigarrow [Rel] Q$

using $\langle eqvt Rel \rangle$

proof(*induct rule: simI*[**where** $C=(xvec, C)$])

case(*cSim* $\alpha Q'$)

from $\langle bn \alpha \#* (xvec, C) \rangle$ **have** $bn \alpha \#* xvec$ **and** $bn \alpha \#* C$ **by** *simp+*

obtain $p::name prm$ **where** $(p \cdot xvec) \#* \Psi$ **and** $(p \cdot xvec) \#* P$ **and** $(p \cdot xvec)$
 $\#* Q$

and $(p \cdot xvec) \#* \alpha$ **and** S : $set p \subseteq set xvec \times set(p \cdot xvec)$

and *distinctPerm p*

by(*rule name-list-avoiding*[**where** $c=(\Psi, P, Q, \alpha)$ **and** $xvec=xvec$]) *auto*

show *?case*

proof(*cases rule: actionCases*[**where** $\alpha=\alpha$])

case(*cInput M N*)

from $\langle \Psi \triangleright Q \mapsto \alpha \prec Q' \rangle \langle \alpha=M(N) \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot Q) \mapsto (p \cdot (M(N)))$
 $\prec Q'$)

by(*auto intro: semantics.eqvt*)

with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle S$

have $QTrans$: $\Psi \triangleright Q \mapsto (p \cdot M)(p \cdot N) \prec (p \cdot Q')$

by(*simp add: eqts*)

moreover from $\langle (p \cdot xvec) \#* \alpha \rangle$ **have** $(p \cdot (p \cdot xvec)) \#* (p \cdot \alpha)$

by(*simp only: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])

with $\langle distinctPerm p \rangle \langle \alpha=M(N) \rangle$ **have** $xvec \#* (p \cdot M)$ **and** $xvec \#* (p \cdot N)$

by *simp+*

moreover with $QTrans \langle xvec \#* Q \rangle$ **have** $xvec \#* (p \cdot Q')$

by(*metis inputFreshChainDerivative*)

ultimately have $\exists P'. \Psi \triangleright P \mapsto (p \cdot M)(p \cdot N) \prec P' \wedge (\Psi, P', (p \cdot Q'))$
 $\in Rel$

by(*simp add: Sim*)
then obtain P' **where** $PTrans: \Psi \triangleright P \mapsto (p \cdot M) \parallel (p \cdot N) \prec P'$ **and**
 $P'RelQ': (\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from $PTrans$ **have** $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot ((p \cdot M) \parallel (p \cdot N)) \prec P')$
by(*rule semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle S \langle distinctPerm$
 $p \rangle$
have $\Psi \triangleright P \mapsto M \parallel N \prec (p \cdot P')$ **by**(*simp add: eqvts*)
moreover from $P'RelQ' \langle eqvt Rel \rangle$ **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add: eqvt-def*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle S \langle distinctPerm p \rangle$
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show $?thesis$ **using** $\langle \alpha = M \parallel N \rangle$ **by** *blast*
next
case(*cBrInput M N*)
from $\langle \Psi \triangleright Q \mapsto \alpha \prec Q' \rangle \langle \alpha = {}_i M \parallel N \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot Q) \mapsto (p \cdot ({}_i M \parallel N))$
 $\prec Q'$
by(*auto intro: semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle S$
have $QTrans: \Psi \triangleright Q \mapsto {}_i(p \cdot M) \parallel (p \cdot N) \prec (p \cdot Q')$
by(*simp add: eqvts*)
moreover from $\langle (p \cdot xvec) \#* \alpha \rangle$ **have** $(p \cdot (p \cdot xvec)) \#* (p \cdot \alpha)$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle distinctPerm p \rangle \langle \alpha = {}_i M \parallel N \rangle$ **have** $xvec \#* (p \cdot M)$ **and** $xvec \#* (p \cdot N)$
by *simp+*
moreover with $QTrans \langle xvec \#* Q \rangle$ **have** $xvec \#* (p \cdot Q')$ **by**(*metis brinput-FreshChainDerivative*)
ultimately have $\exists P'. \Psi \triangleright P \mapsto {}_i(p \cdot M) \parallel (p \cdot N) \prec P' \wedge (\Psi, P', (p \cdot Q'))$
 $\in Rel$
by(*simp add: Sim*)
then obtain P' **where** $PTrans: \Psi \triangleright P \mapsto {}_i(p \cdot M) \parallel (p \cdot N) \prec P'$ **and**
 $P'RelQ': (\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from $PTrans$ **have** $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot ({}_i(p \cdot M) \parallel (p \cdot N)) \prec P')$
by(*rule semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle S \langle distinctPerm$
 $p \rangle$
have $\Psi \triangleright P \mapsto {}_i M \parallel N \prec (p \cdot P')$ **by**(*simp add: eqvts*)
moreover from $P'RelQ' \langle eqvt Rel \rangle$ **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add: eqvt-def*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle S \langle distinctPerm p \rangle$
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show $?thesis$ **using** $\langle \alpha = {}_i M \parallel N \rangle$ **by** *blast*
next
case(*cOutput M yvec N*)
from $\langle distinct(bn \alpha) \rangle \langle bn \alpha \#* subject \alpha \rangle \langle \alpha = M \parallel ({}_{\nu} yvec) \parallel N \rangle$ **have** *distinct*
 $yvec$ **and** $yvec \#* M$ **by** *simp+*
from $\langle \Psi \triangleright Q \mapsto \alpha \prec Q' \rangle \langle \alpha = M \parallel ({}_{\nu} yvec) \parallel N \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot Q) \mapsto (p$

$\cdot (M(\nu*yvec)\langle N \rangle \prec Q')$
by(*auto intro: semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle S$
have $QTrans: \Psi \triangleright Q \mapsto (p \cdot M)(\nu*(p \cdot yvec))\langle (p \cdot N) \rangle \prec (p \cdot Q')$
by(*simp add: eqts*)
with $S \langle bn \alpha \#* xvec \rangle \langle (p \cdot xvec) \#* \alpha \rangle \langle \alpha = M(\nu*yvec)\langle N \rangle \rangle$ **have** $\Psi \triangleright Q \mapsto (p \cdot M)(\nu*yvec)\langle (p \cdot N) \rangle \prec (p \cdot Q')$
by *simp*
moreover from $\langle (p \cdot xvec) \#* \alpha \rangle$ **have** $(p \cdot (p \cdot xvec)) \#* (p \cdot \alpha)$
by(*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle distinctPerm p \rangle \langle \alpha = M(\nu*yvec)\langle N \rangle \rangle$ **have** $xvec \#* (p \cdot M)$ **and** $xvec \#* (p \cdot N)$ **and** $xvec \#* (p \cdot yvec)$ **by** *simp+*
moreover with $QTrans \langle xvec \#* Q \rangle \langle distinct yvec \rangle \langle yvec \#* M \rangle$ **have** $xvec \#* (p \cdot Q')$
apply(*drule-tac outputFreshChainDerivative(2)*)
by(*auto simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle yvec \#* M \rangle S \langle bn \alpha \#* xvec \rangle \langle (p \cdot xvec) \#* \alpha \rangle \langle \alpha = M(\nu*yvec)\langle N \rangle \rangle$
 $\langle distinctPerm p \rangle$
have $yvec \#* (p \cdot M)$ **by**(*subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, symmetric, where pi=p]*) *simp*
ultimately have $\exists P'. \Psi \triangleright P \mapsto (p \cdot M)(\nu*yvec)\langle (p \cdot N) \rangle \prec P' \wedge (\Psi, P', (p \cdot Q')) \in Rel$
using $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* xvec \rangle \langle bn \alpha \#* C \rangle \langle yvec \#* M \rangle \langle \alpha = M(\nu*yvec)\langle N \rangle \rangle \langle distinct yvec \rangle$
by(*simp add: Sim*)
then obtain P' **where** $PTrans: \Psi \triangleright P \mapsto (p \cdot M)(\nu*yvec)\langle (p \cdot N) \rangle \prec P'$
and $P'RelQ': (\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from $PTrans$ **have** $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot ((p \cdot M)(\nu*yvec)\langle (p \cdot N) \rangle \prec P'))$
by(*rule semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle S \langle distinctPerm p \rangle \langle bn \alpha \#* xvec \rangle \langle (p \cdot xvec) \#* \alpha \rangle \langle \alpha = M(\nu*yvec)\langle N \rangle \rangle$
have $\Psi \triangleright P \mapsto M(\nu*yvec)\langle N \rangle \prec (p \cdot P')$ **by**(*simp add: eqts*)
moreover from $P'RelQ' \langle eqvt Rel \rangle$ **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by(*auto simp add: eqvt-def*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle S \langle distinctPerm p \rangle$
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show *?thesis* **using** $\langle \alpha = M(\nu*yvec)\langle N \rangle \rangle$ **by** *blast*
next
case(*cBrOutput M yvec N*)
from $\langle distinct(bn \alpha) \rangle \langle bn \alpha \#* subject \alpha \rangle \langle \alpha = {}_iM(\nu*yvec)\langle N \rangle \rangle$ **have** *distinct yvec*
and $yvec \#* M$ **by** *simp+*
from $\langle \Psi \triangleright Q \mapsto \alpha \prec Q' \rangle \langle \alpha = {}_iM(\nu*yvec)\langle N \rangle \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot Q) \mapsto (p \cdot ({}_iM(\nu*yvec)\langle N \rangle \prec Q'))$
by(*auto intro: semantics.eqvt*)
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle S$
have $QTrans: \Psi \triangleright Q \mapsto {}_i(p \cdot M)(\nu*(p \cdot yvec))\langle (p \cdot N) \rangle \prec (p \cdot Q')$
by(*simp add: eqts*)

with $S \langle bn \ \alpha \ \#* \ xvec \rangle \langle (p \cdot xvec) \ \#* \ \alpha \rangle \langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle$ **have** $\Psi \triangleright Q$
 $\mapsto_i (p \cdot M)(\nu * yvec) \langle (p \cdot N) \rangle \prec (p \cdot Q')$
by *simp*
moreover from $\langle (p \cdot xvec) \ \#* \ \alpha \rangle$ **have** $(p \cdot (p \cdot xvec)) \ \#* \ (p \cdot \alpha)$
by (*simp only: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle distinctPerm \ p \rangle \langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle$ **have** $xvec \ \#* \ (p \cdot M)$ **and** $xvec \ \#*$
 $(p \cdot N)$ **and** $xvec \ \#* \ (p \cdot yvec)$ **by** *simp+*
moreover with $QTrans \ \langle xvec \ \#* \ Q \rangle \langle distinct \ yvec \rangle \langle yvec \ \#* \ M \rangle$ **have** $xvec \ \#*$
 $(p \cdot Q')$
apply (*drule-tac broutputFreshChainDerivative(2)*)
by (*auto simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
moreover from $\langle yvec \ \#* \ M \rangle S \langle bn \ \alpha \ \#* \ xvec \rangle \langle (p \cdot xvec) \ \#* \ \alpha \rangle \langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle$
 $\langle distinctPerm \ p \rangle$
have $yvec \ \#* \ (p \cdot M)$ **by** (*subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst,*
symmetric, where pi=p]) *simp*
ultimately have $\exists P'. \Psi \triangleright P \mapsto_i (p \cdot M)(\nu * yvec) \langle (p \cdot N) \rangle \prec P' \wedge (\Psi, P',$
 $(p \cdot Q')) \in Rel$
using $\langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle bn \ \alpha \ \#* \ xvec \rangle \langle bn \ \alpha \ \#* \ C \rangle \langle yvec$
 $\ \#* \ M \rangle \langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle \langle distinct \ yvec \rangle$
by (*simp add: Sim*)
then obtain P' **where** $PTrans: \Psi \triangleright P \mapsto_i (p \cdot M)(\nu * yvec) \langle (p \cdot N) \rangle \prec P'$
and $P'RelQ': (\Psi, P', (p \cdot Q')) \in Rel$
by *blast*
from $PTrans$ **have** $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto_i (p \cdot (i(p \cdot M)(\nu * yvec) \langle (p \cdot N) \rangle \prec$
 $P'))$
by (*rule semantics.eqvt*)
with $\langle xvec \ \#* \ \Psi \rangle \langle (p \cdot xvec) \ \#* \ \Psi \rangle \langle xvec \ \#* \ P \rangle \langle (p \cdot xvec) \ \#* \ P \rangle S \langle distinctPerm$
 $p \rangle \langle bn \ \alpha \ \#* \ xvec \rangle \langle (p \cdot xvec) \ \#* \ \alpha \rangle \langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle$
have $\Psi \triangleright P \mapsto_i M(\nu * yvec) \langle N \rangle \prec (p \cdot P')$ **by** (*simp add: eqvts*)
moreover from $P'RelQ' \langle eqvt \ Rel \rangle$ **have** $(p \cdot \Psi, p \cdot P', p \cdot p \cdot Q') \in Rel$
by (*auto simp add: eqvt-def*)
with $\langle xvec \ \#* \ \Psi \rangle \langle (p \cdot xvec) \ \#* \ \Psi \rangle S \langle distinctPerm \ p \rangle$
have $(\Psi, p \cdot P', Q') \in Rel$ **by** *simp*
ultimately show *?thesis* **using** $\langle \alpha =_i M(\nu * yvec) \langle N \rangle \rangle$ **by** *blast*
next
case *cTau*
from $\langle \Psi \triangleright Q \mapsto_i \alpha \prec Q' \rangle \langle \alpha = \tau \rangle \langle xvec \ \#* \ Q \rangle$ **have** $xvec \ \#* \ Q'$
by (*blast dest: tauFreshChainDerivative*)
with $\langle \Psi \triangleright Q \mapsto_i \alpha \prec Q' \rangle \langle \alpha = \tau \rangle$
show *?thesis*
using *Sim* **by** *simp*
qed
qed

lemma *simIFresh[consumes 4, case-names cSim]:*

fixes $\Psi \quad :: 'b$
and $P \quad :: ('a, 'b, 'c) \ psi$
and $Rel \quad :: ('b \times ('a, 'b, 'c) \ psi \times ('a, 'b, 'c) \ psi) \ set$
and $Q \quad :: ('a, 'b, 'c) \ psi$

```

and  $x$  :: name
and  $C$  :: 'd::fs-name

assumes Eqvt: eqvt Rel
and  $x \# \Psi$ 
and  $x \# P$ 
and  $x \# Q$ 
and  $\bigwedge \alpha Q'. [\Psi \triangleright Q \mapsto \alpha \prec Q'; \text{bn } \alpha \#* P; \text{bn } \alpha \#* Q; \text{bn } \alpha \#* \Psi;$ 
     $\text{bn } \alpha \#* \text{subject } \alpha; \text{distinct}(\text{bn } \alpha); \text{bn } \alpha \#* C; x \# \alpha; x \# Q'] \implies$ 
     $\exists P'. \Psi \triangleright P \mapsto \alpha \prec P' \wedge (\Psi, P', Q') \in \text{Rel}$ 

shows  $\Psi \triangleright P \rightsquigarrow[\text{Rel}] Q$ 
using assms
by (auto intro: simIChainFresh[where xvec=[x] and C=C])

lemma simE:
fixes  $F$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\text{Rel}$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$  :: ('a, 'b, 'c) psi

assumes  $\Psi \triangleright P \rightsquigarrow[\text{Rel}] Q$ 

shows  $\bigwedge \alpha Q'. [\Psi \triangleright Q \mapsto \alpha \prec Q'; \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P] \implies \exists P'. \Psi \triangleright P \mapsto \alpha$ 
 $\prec P' \wedge (\Psi, P', Q') \in \text{Rel}$ 
using assms
by(auto simp add: simulation-def)

lemma simClosedAux:
fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $\text{Rel}$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$  :: ('a, 'b, 'c) psi
and  $p$  :: name prm

assumes EqvtRel: eqvt Rel
and PSimQ:  $\Psi \triangleright P \rightsquigarrow[\text{Rel}] Q$ 

shows  $(p \cdot \Psi) \triangleright (p \cdot P) \rightsquigarrow[\text{Rel}] (p \cdot Q)$ 
using EqvtRel
proof(induct rule: simI[of - - - (\Psi, P, p)])
case(cSim  $\alpha Q'$ )
from  $\langle p \cdot \Psi \triangleright p \cdot Q \mapsto \alpha \prec Q' \rangle$ 
have  $(\text{rev } p \cdot p \cdot \Psi) \triangleright (\text{rev } p \cdot p \cdot Q) \mapsto (\text{rev } p \cdot (\alpha \prec Q'))$ 
by(blast dest: semantics.eqvt)
then have  $\Psi \triangleright Q \mapsto (\text{rev } p \cdot \alpha) \prec (\text{rev } p \cdot Q')$ 
by(simp add: eqvts)
moreover with  $\langle \text{bn } \alpha \#* (\Psi, P, p) \rangle$  have  $\text{bn } \alpha \#* \Psi$  and  $\text{bn } \alpha \#* P$  and  $\text{bn } \alpha$ 
 $\#* p$  by simp+

```

```

ultimately obtain  $P'$  where  $PTrans: \Psi \triangleright P \mapsto (rev\ p \cdot \alpha) \prec P'$ 
and  $P'RelQ': (\Psi, P', rev\ p \cdot Q') \in Rel$ 
using  $PSimQ$ 
by(force dest: simE freshChainPermSimp simp add: eqvts)
from  $PTrans$  have  $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto (p \cdot ((rev\ p \cdot \alpha) \prec P'))$ 
by(rule semantics.eqt)
with  $\langle bn\ \alpha\ \#*\ p \rangle$  have  $(p \cdot \Psi) \triangleright (p \cdot P) \mapsto \alpha \prec (p \cdot P')$ 
by(simp add: eqvts freshChainPermSimp)
moreover from  $P'RelQ' EqvRel$  have  $(p \cdot (\Psi, P', rev\ p \cdot Q')) \in Rel$ 
by(simp only: eqv-def)
then have  $(p \cdot \Psi, p \cdot P', Q') \in Rel$  by simp
ultimately show ?case by blast
qed

```

lemma *simClosed*:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $Rel$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$  :: ('a, 'b, 'c) psi
and  $p$  :: name prm

```

assumes $EqvRel$: $eqv\ Rel$

```

shows  $\Psi \triangleright P \rightsquigarrow[Rel] Q \implies (p \cdot \Psi) \triangleright (p \cdot P) \rightsquigarrow[Rel] (p \cdot Q)$ 
and  $P \rightsquigarrow[Rel] Q \implies (p \cdot P) \rightsquigarrow[Rel] (p \cdot Q)$ 
using  $EqvRel$ 
by(force dest: simClosedAux simp add: permBottom)+

```

lemma *reflexive*:

```

fixes  $Rel$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi

```

assumes $\{(\Psi, P, P) \mid \Psi\ P.\ True\} \subseteq Rel$

```

shows  $\Psi \triangleright P \rightsquigarrow[Rel] P$ 
using assms
by(auto simp add: simulation-def)

```

lemma *transitive*:

```

fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $Rel$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$  :: ('a, 'b, 'c) psi
and  $Rel'$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $R$  :: ('a, 'b, 'c) psi
and  $Rel''$  :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set

```

assumes $PSimQ$: $\Psi \triangleright P \rightsquigarrow[Rel] Q$

and $QSimR: \Psi \triangleright Q \rightsquigarrow[Rel'] R$
and $Eqvt: eqvt Rel''$
and $Set: \{(\Psi, P, R) \mid \Psi P R. \exists Q. (\Psi, P, Q) \in Rel \wedge (\Psi, Q, R) \in Rel'\} \subseteq Rel''$

shows $\Psi \triangleright P \rightsquigarrow[Rel''] R$
using $\langle eqvt Rel'' \rangle$
proof(*induct rule: simI*[**where** $C=Q$])
case($cSim \alpha R'$)
from $QSimR \langle \Psi \triangleright R \mapsto \alpha \prec R' \rangle \langle (bn \alpha) \#* \Psi \rangle \langle (bn \alpha) \#* Q \rangle$
obtain Q' **where** $QTrans: \Psi \triangleright Q \mapsto \alpha \prec Q'$ **and** $Q'Rel'R': (\Psi, Q', R') \in Rel'$
by(*blast dest: simE*)
from $PSimQ QTrans \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle$
obtain P' **where** $PTrans: \Psi \triangleright P \mapsto \alpha \prec P'$ **and** $P'RelQ': (\Psi, P', Q') \in Rel$
by(*blast dest: simE*)
with $PTrans Q'Rel'R' P'RelQ' Set$
show $?case$ **by** *blast*
qed

lemma *statEqSim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$
and $Q :: ('a, 'b, 'c) psi$
and $\Psi' :: 'b$

assumes $PSimQ: \Psi \triangleright P \rightsquigarrow[Rel] Q$
and $eqvt Rel'$
and $\Psi \simeq \Psi'$
and $C1: \bigwedge \Psi'' R S \Psi'''. \llbracket (\Psi'', R, S) \in Rel; \Psi'' \simeq \Psi''' \rrbracket \implies (\Psi''', R, S) \in Rel'$

shows $\Psi' \triangleright P \rightsquigarrow[Rel'] Q$
using $\langle eqvt Rel' \rangle$
proof(*induct rule: simI*[*of* - - - Ψ])
case($cSim \alpha Q'$)
from $\langle \Psi' \triangleright Q \mapsto \alpha \prec Q' \rangle \langle \Psi \simeq \Psi' \rangle$
have $\Psi \triangleright Q \mapsto \alpha \prec Q'$ **by**(*metis statEqTransition AssertionStatEqSym*)
with $PSimQ \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle$
obtain P' **where** $\Psi \triangleright P \mapsto \alpha \prec P'$ **and** $(\Psi, P', Q') \in Rel$
by(*blast dest: simE*)

from $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle \Psi \simeq \Psi' \rangle$ **have** $\Psi' \triangleright P \mapsto \alpha \prec P'$
by(*rule statEqTransition*)
moreover from $\langle (\Psi, P', Q') \in Rel \rangle \langle \Psi \simeq \Psi' \rangle$ **have** $(\Psi', P', Q') \in Rel'$
by(*rule C1*)
ultimately show $?case$ **by** *blast*
qed

lemma *monotonic*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{psi}$ 
  and  $A :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{set}$ 
  and  $Q :: ('a, 'b, 'c) \text{psi}$ 
  and  $B :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{set}$ 

assumes  $\Psi \triangleright P \rightsquigarrow[A] Q$ 
  and  $A \subseteq B$ 

shows  $\Psi \triangleright P \rightsquigarrow[B] Q$ 
  using assms
  by(simp (no-asm) add: simulation-def (auto dest: simE))

```

end

end

theory *Bisimulation*

imports *Simulation*

begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Bisimulation* from [1].

context *env* **begin**

lemma *monoCoinduct*: $\bigwedge x y xa xb xc P Q \Psi.$

$$\begin{aligned}
 & x \leq y \implies \\
 & (\Psi \triangleright Q \rightsquigarrow[\{(xc, xb, xa). x xc xb xa\}] P) \longrightarrow \\
 & (\Psi \triangleright Q \rightsquigarrow[\{(xb, xa, xc). y xb xa xc\}] P)
 \end{aligned}$$

apply(*rule impI*)

apply(*rule monotonic*)

by(*auto dest: le-funE*)

coinductive-set *bisim* :: $('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{set}$

where

step: $\llbracket (\text{insertAssertion } (\text{extractFrame } P)) \Psi \simeq_F (\text{insertAssertion } (\text{extractFrame } Q) \Psi) \rrbracket;$

$\Psi \triangleright P \rightsquigarrow[\text{bisim}] Q;$

$\forall \Psi'. (\Psi \otimes \Psi', P, Q) \in \text{bisim}; (\Psi, Q, P) \in \text{bisim} \rrbracket \implies (\Psi, P, Q) \in \text{bisim}$

monos *monoCoinduct*

abbreviation

bisimJudge $(- \triangleright - \rightsquigarrow - [70, 70, 70] 65)$ **where** $\Psi \triangleright P \rightsquigarrow Q \equiv (\Psi, P, Q) \in \text{bisim}$

abbreviation

bisimNilJudge $(- \rightsquigarrow - [70, 70] 65)$ **where** $P \rightsquigarrow Q \equiv \text{SBottom}' \triangleright P \rightsquigarrow Q$

lemma *bisimCoinductAux*[*consumes 1*]:

fixes $F :: 'b$

and $P :: ('a, 'b, 'c) \text{psi}$

and $Q :: ('a, 'b, 'c) \text{psi}$

and $X :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F$
 $\text{insertAssertion } (\text{extractFrame } Q) \Psi \wedge$
 $(\Psi \triangleright P \rightsquigarrow[(X \cup \text{bisim})] Q) \wedge$
 $(\forall \Psi'. (\Psi \otimes \Psi', P, Q) \in X \vee (\Psi \otimes \Psi', P, Q) \in$
 $\text{bisim}) \wedge$
 $((\Psi, Q, P) \in X \vee (\Psi, Q, P) \in \text{bisim})$

shows $(\Psi, P, Q) \in \text{bisim}$

proof –
have $X \cup \text{bisim} = \{(\Psi, P, Q). (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in \text{bisim}\}$ **by auto**
with *assms show ?thesis*
by *coinduct simp*

qed

lemma *bisimCoinduct*[*consumes 1, case-names cStatEq cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c) \text{psi}$
and $Q :: ('a, 'b, 'c) \text{psi}$
and $X :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies \text{insertAssertion } (\text{extractFrame } R) \Psi' \simeq_F$
 $\text{insertAssertion } (\text{extractFrame } S) \Psi'$
and $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies \Psi' \triangleright R \rightsquigarrow[(X \cup \text{bisim})] S$
and $\bigwedge \Psi' R S \Psi''. (\Psi', R, S) \in X \implies (\Psi' \otimes \Psi'', R, S) \in X \vee (\Psi' \otimes \Psi'', R,$
 $S) \in \text{bisim}$
and $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies (\Psi', S, R) \in X \vee (\Psi', S, R) \in \text{bisim}$

shows $(\Psi, P, Q) \in \text{bisim}$

proof –
have $X \cup \text{bisim} = \{(\Psi, P, Q). (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in \text{bisim}\}$ **by auto**
with *assms show ?thesis*
by *coinduct simp*

qed

lemma *bisimWeakCoinductAux*[*consumes 1*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) \text{psi}$
and $Q :: ('a, 'b, 'c) \text{psi}$
and $X :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F$
 $\text{insertAssertion } (\text{extractFrame } Q) \Psi \wedge$
 $\Psi \triangleright P \rightsquigarrow[X] Q \wedge$
 $(\forall \Psi'. (\Psi \otimes \Psi', P, Q) \in X) \wedge (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in \text{bisim}$
using *assms*
by(*coinduct rule: bisimCoinductAux*) (*blast intro: monotonic*)

lemma *bisimWeakCoinduct*[*consumes 1, case-names cStatEq cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c) \text{psi}$
and $Q :: ('a, 'b, 'c) \text{psi}$
and $X :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F$
 $\text{insertAssertion } (\text{extractFrame } Q) \Psi$
and $\bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow[X] Q$
and $\bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in X \implies (\Psi \otimes \Psi', P, Q) \in X$
and $\bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in \text{bisim}$
proof –
have $X \cup \text{bisim} = \{(\Psi, P, Q). (\Psi, P, Q) \in X \vee (\Psi, P, Q) \in \text{bisim}\}$ **by** *auto*
with *assms* **show** *?thesis*
by(*coinduct rule: bisimCoinduct*) (*blast intro: monotonic*)+
qed

lemma *bisimE*:
fixes $P :: ('a, 'b, 'c) \text{psi}$
and $Q :: ('a, 'b, 'c) \text{psi}$
and $\Psi :: 'b$
and $\Psi' :: 'b$

assumes $(\Psi, P, Q) \in \text{bisim}$

shows $\text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F \text{insertAssertion } (\text{extractFrame } Q)$
 Ψ
and $\Psi \triangleright P \rightsquigarrow[\text{bisim}] Q$
and $(\Psi \otimes \Psi', P, Q) \in \text{bisim}$
and $(\Psi, Q, P) \in \text{bisim}$
using *assms*
by(*auto simp add: intro: bisim.cases*)

lemma *bisimI*:
fixes $P :: ('a, 'b, 'c) \text{psi}$
and $Q :: ('a, 'b, 'c) \text{psi}$
and $\Psi :: 'b$

assumes $\text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F \text{insertAssertion } (\text{extractFrame } Q)$
 Ψ
and $\Psi \triangleright P \rightsquigarrow[\text{bisim}] Q$

```

and  $\forall \Psi'. (\Psi \otimes \Psi', P, Q) \in \text{bisim}$ 
and  $(\Psi, Q, P) \in \text{bisim}$ 

shows  $(\Psi, P, Q) \in \text{bisim}$ 
using assms
by(auto intro: bisim.step)

lemma bisimReflexive:
fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) \text{ psi}$ 

shows  $\Psi \triangleright P \sim P$ 
proof –
let  $?X = \{(\Psi, P, P) \mid \Psi P. \text{True}\}$ 
have  $(\Psi, P, P) \in ?X$  by simp
then show ?thesis
by(coinduct rule: bisimWeakCoinduct, auto intro: reflexive)
qed

lemma bisimClosed:
fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) \text{ psi}$ 
and  $Q :: ('a, 'b, 'c) \text{ psi}$ 
and  $p :: \text{name prm}$ 

assumes PBisimQ:  $\Psi \triangleright P \sim Q$ 

shows  $(p \cdot \Psi) \triangleright (p \cdot P) \sim (p \cdot Q)$ 
proof –
let  $?X = \{(p \cdot \Psi, p \cdot P, p \cdot Q) \mid (p::\text{name prm}) \Psi P Q. \Psi \triangleright P \sim Q\}$ 
from PBisimQ have  $(p \cdot \Psi, p \cdot P, p \cdot Q) \in ?X$  by blast
then show ?thesis
proof(coinduct rule: bisimWeakCoinduct)
case(cStatEq  $\Psi P Q$ )
have  $\bigwedge \Psi P Q (p::\text{name prm}). \text{insertAssertion} (\text{extractFrame } P) \Psi \simeq_F \text{insertAssertion} (\text{extractFrame } Q) \Psi \implies$ 
 $\text{insertAssertion} (\text{extractFrame}(p \cdot P)) (p \cdot \Psi) \simeq_F \text{insertAssertion} (\text{extractFrame}(p \cdot Q)) (p \cdot \Psi)$ 
by(drule FrameStatEqClosed) (simp add: eqvts)

with  $\langle (\Psi, P, Q) \in ?X \rangle$  show ?case by(blast dest: bisimE)
next
case(cSim  $\Psi P Q$ )
{
fix  $p :: \text{name prm}$ 
fix  $\Psi P Q$ 
have eqvt  $?X$ 
apply(clarsimp simp add: eqvt-def)

```

```

    by (metis (no-types, opaque-lifting) pt-name2)
  moreover assume  $\Psi \triangleright P \rightsquigarrow[bisim] Q$ 
  then have  $\Psi \triangleright P \rightsquigarrow[?X] Q$ 
    apply (rule monotonic[where A=bisim], auto)
    by (rule exI[where x=[]::name prm]) auto
  ultimately have  $((p::name prm) \cdot \Psi) \triangleright (p \cdot P) \rightsquigarrow[?X] (p \cdot Q)$ 
    by (rule simClosed)
}
with  $\langle \Psi, P, Q \rangle \in ?X$  show ?case
  by (blast dest: bisimE)
next
case (cExt  $\Psi P Q \Psi'$ )
{
  fix p :: name prm
  fix  $\Psi P Q \Psi'$ 
  assume  $\forall \Psi'. (\Psi \otimes \Psi', P, Q) \in bisim$ 
  then have  $((p \cdot \Psi) \otimes \Psi', p \cdot P, p \cdot Q) \in ?X$ 
    apply (clarisimp)
    apply (rule exI[where x=p])
    apply (rule exI[where x= $\Psi \otimes (rev p \cdot \Psi')$ ])
    by (auto simp add: eqts)
}
with  $\langle \Psi, P, Q \rangle \in ?X$  show ?case
  by (blast dest: bisimE)
next
case (cSym  $\Psi P Q$ )
then show ?case
  by (blast dest: bisimE)
qed
qed

```

lemma *bisimEqvt[simp]*:
 shows *eqvt bisim*
 by (auto simp add: eqvt-def bisimClosed)

lemma *statEqBisim*:
 fixes $\Psi :: 'b$
 and $P :: ('a, 'b, 'c) psi$
 and $Q :: ('a, 'b, 'c) psi$
 and $\Psi' :: 'b$

assumes $\Psi \triangleright P \sim Q$
 and $\Psi \simeq \Psi'$

shows $\Psi' \triangleright P \sim Q$

proof –

let $?X = \{(\Psi', P, Q) \mid \Psi P Q \Psi'. \Psi \triangleright P \sim Q \wedge \Psi \simeq \Psi'\}$
 from $\langle \Psi \triangleright P \sim Q \rangle \langle \Psi \simeq \Psi' \rangle$ have $(\Psi', P, Q) \in ?X$ by auto
 then show ?thesis

```

proof(coinduct rule: bisimCoinduct)
  case(cStatEq  $\Psi' P Q$ )
    from  $\langle \Psi', P, Q \rangle \in ?X$  obtain  $\Psi$  where  $\Psi \triangleright P \sim Q$  and  $\Psi \simeq \Psi'$ 
      by auto
      from  $\langle \Psi \triangleright P \sim Q \rangle$  have PeqQ: insertAssertion (extractFrame P)  $\Psi \simeq_F$ 
insertAssertion (extractFrame Q)  $\Psi$ 
      by(rule bisimE)

    obtain  $A_P \Psi_P$  where FrP: extractFrame P =  $\langle A_P, \Psi_P \rangle$  and  $A_P \#* \Psi$  and
 $A_P \#* \Psi'$ 
      by(rule freshFrame[where  $C=(\Psi, \Psi')$ ]) auto
      obtain  $A_Q \Psi_Q$  where FrQ: extractFrame Q =  $\langle A_Q, \Psi_Q \rangle$  and  $A_Q \#* \Psi$  and
 $A_Q \#* \Psi'$ 
      by(rule freshFrame[where  $C=(\Psi, \Psi')$ ]) auto

    from PeqQ FrP FrQ  $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle \Psi \simeq \Psi' \rangle$ 
    have  $\langle A_P, \Psi' \otimes \Psi_P \rangle \simeq_F \langle A_Q, \Psi' \otimes \Psi_Q \rangle$ 
      by simp (metis frameIntComposition FrameStatEqTrans FrameStatEqSym)
    with FrP FrQ  $\langle A_P \#* \Psi' \rangle \langle A_Q \#* \Psi' \rangle$  show ?case by simp
  next
    case(cSim  $\Psi' P Q$ )
      from  $\langle \Psi', P, Q \rangle \in ?X$  obtain  $\Psi$  where  $\Psi \triangleright P \sim Q$  and  $\Psi \simeq \Psi'$ 
        by auto
        from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \triangleright P \rightsquigarrow[bisim] Q$  by(blast dest: bisimE)
        moreover have eqvt  $?X$ 
          by(auto simp add: eqvt-def) (metis bisimClosed AssertionStatEqClosed)
        then have eqvt( $?X \cup bisim$ ) by auto
        moreover note  $\langle \Psi \simeq \Psi' \rangle$ 
        moreover have  $\bigwedge \Psi P Q \Psi'. [\Psi \triangleright P \sim Q; \Psi \simeq \Psi'] \implies (\Psi', P, Q) \in ?X \cup$ 
bisim
          by auto
          ultimately show ?case
          by(rule statEqSim)
      next
        case(cExt  $\Psi' P Q \Psi''$ )
          from  $\langle \Psi', P, Q \rangle \in ?X$  obtain  $\Psi$  where  $\Psi \triangleright P \sim Q$  and  $\Psi \simeq \Psi'$ 
            by auto
            from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \otimes \Psi'' \triangleright P \sim Q$  by(rule bisimE)
            moreover from  $\langle \Psi \simeq \Psi' \rangle$  have  $\Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi''$  by(rule Composition)
            ultimately show ?case by blast
          next
            case(cSym  $\Psi' P Q$ )
              from  $\langle \Psi', P, Q \rangle \in ?X$  obtain  $\Psi$  where  $\Psi \triangleright P \sim Q$  and  $\Psi \simeq \Psi'$ 
                by auto
                from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \triangleright Q \sim P$  by(rule bisimE)
                then show ?case using  $\langle \Psi \simeq \Psi' \rangle$  by auto
            qed
          qed

```

```

lemma bisimTransitive:
  fixes  $\Psi :: 'b$ 
    and  $P :: ('a, 'b, 'c)$  psi
    and  $Q :: ('a, 'b, 'c)$  psi
    and  $R :: ('a, 'b, 'c)$  psi

  assumes  $PQ: \Psi \triangleright P \sim Q$ 
    and  $QR: \Psi \triangleright Q \sim R$ 

  shows  $\Psi \triangleright P \sim R$ 
  proof -
    let  $?X = \{(\Psi, P, R) \mid \Psi P Q R. \Psi \triangleright P \sim Q \wedge \Psi \triangleright Q \sim R\}$ 
    from  $PQ QR$  have  $(\Psi, P, R) \in ?X$  by auto
    then show ?thesis
    proof(coinduct rule: bisimCoinduct)
      case(cStatEq  $\Psi P R$ )
        then show ?case by(blast dest: bisimE FrameStatEqTrans)
    next
      case(cSim  $\Psi P R$ )
        {
          fix  $\Psi P Q R$ 
          assume  $\Psi \triangleright P \rightsquigarrow[bisim] Q$  and  $\Psi \triangleright Q \rightsquigarrow[bisim] R$ 
          moreover have eqvt  $?X$ 
            by(force simp add: eqvt-def dest: bisimClosed)
          with bisimEqvt have eqvt  $(?X \cup bisim)$  by blast
          moreover have  $?X \subseteq ?X \cup bisim$  by auto
          ultimately have  $\Psi \triangleright P \rightsquigarrow[(?X \cup bisim)] R$ 
            by(force intro: transitive)
        }
      with  $(\Psi, P, R) \in ?X$  show ?case
        by(blast dest: bisimE)
    next
      case(cExt  $\Psi P R \Psi'$ )
        then show ?case by(blast dest: bisimE)
    next
      case(cSym  $\Psi P R$ )
        then show ?case by(blast dest: bisimE)
    qed
  qed

```

lemma *weakTransitiveCoinduct*[*case-names cStatEq cSim cExt cSym, case-conclusion bisim step, consumes 2*]:

```

  fixes  $\Psi :: 'b$ 
    and  $P :: ('a, 'b, 'c)$  psi
    and  $Q :: ('a, 'b, 'c)$  psi
    and  $X :: ('b \times ('a, 'b, 'c)$  psi  $\times ('a, 'b, 'c)$  psi) set

```

```

  assumes  $p: (\Psi, P, Q) \in X$ 
    and Eqvt: eqvt  $X$ 

```

and $rStatEq: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies insertAssertion (extractFrame P) \Psi$
 $\simeq_F insertAssertion (extractFrame Q) \Psi$
and $rSim: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow [(\{\Psi, P, Q\} \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge$
 $\Psi \triangleright P \sim P' \wedge$

$$(\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q)] Q$$
and $rExt: \bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in X \implies (\Psi \otimes \Psi', P, Q) \in X$
and $rSym: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies (\Psi, Q, P) \in X$

shows $\Psi \triangleright P \sim Q$

proof –

let $?X = \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$

from p **have** $(\Psi, P, Q) \in ?X$

by(*blast intro: bisimReflexive*)

then show $?thesis$

proof(*coinduct rule: bisimWeakCoinduct*)

case($cStatEq \Psi P Q$)

then show $?case$

by(*blast dest: rStatEq bisimE FrameStatEqTrans*)

next

case($cSim \Psi P Q$)

{

fix $\Psi P P' Q' Q$

assume $\Psi \triangleright P \rightsquigarrow [bisim] P'$

moreover assume $P' Rel Q': (\Psi, P', Q') \in X$

then have $\Psi \triangleright P' \rightsquigarrow [?X] Q'$ **by**(*rule rSim*)

moreover from $\langle eqvt X \rangle P' Rel Q'$ **have** $eqvt ?X$

apply(*clarsimp simp add: eqvt-def*)

apply(*drule bisimClosed*)

apply(*drule bisimClosed*)

apply(*rule exI*)

apply(*rule conjI*)

apply *assumption*

apply(*rule exI*)

by *auto*

ultimately have $\Psi \triangleright P \rightsquigarrow [?X] Q'$

by(*force intro: transitive dest: bisimTransitive*)

moreover assume $\Psi \triangleright Q' \rightsquigarrow [bisim] Q$

ultimately have $\Psi \triangleright P \rightsquigarrow [?X] Q$ **using** $\langle eqvt ?X \rangle$

by(*force intro: transitive dest: bisimTransitive*)

}

with $\langle (\Psi, P, Q) \in ?X \rangle$ **show** $?case$

by(*blast dest: bisimE*)

next

case($cExt \Psi P Q \Psi'$)

then show $?case$ **by**(*blast dest: bisimE intro: rExt*)

next

case($cSym \Psi P Q$)

```

    then show ?case by(blast dest: bisimE intro: rSym)
  qed
qed

```

lemma *weakTransitiveCoinduct* [case-names *cStatEq cSim cExt cSym*, case-conclusion *bisim step*, consumes 2]:

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $X :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

  assumes  $p: (\Psi, P, Q) \in X$ 
  and  $Eqvt: eqvt X$ 
  and  $rStatEq: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies insertAssertion (extractFrame P) \Psi$ 
 $\simeq_F insertAssertion (extractFrame Q) \Psi$ 
  and  $rSim: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow [(\{\Psi, P, Q\} \mid \Psi P P' Q' Q.$ 
 $\Psi \triangleright P \sim P' \wedge$ 
 $\Psi \triangleright Q' \sim Q)] Q$ 
 $(\Psi, P', Q') \in X \wedge$ 
 $\Psi \triangleright Q' \sim Q)] Q$ 
  and  $rExt: \bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in X \implies (\Psi \otimes \Psi', P, Q) \in X$ 
  and  $rSym: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies$ 
 $(\Psi, Q, P) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P',$ 
 $Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 

```

shows $\Psi \triangleright P \sim Q$

proof –

```

  let ?X = {(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q'
  \sim Q}

```

```

  from  $p$  have  $(\Psi, P, Q) \in ?X$ 

```

```

  by(blast intro: bisimReflexive)

```

```

  then show ?thesis

```

```

  proof(coinduct rule: bisimWeakCoinduct)

```

```

    case(cStatEq  $\Psi P Q$ )

```

```

    then show ?case

```

```

      by(blast dest: rStatEq bisimE FrameStatEqTrans)

```

```

  next

```

```

    case(cSim  $\Psi P Q$ )

```

```

    {

```

```

      fix  $\Psi P P' Q' Q$ 

```

```

      assume  $\Psi \triangleright P \rightsquigarrow [bisim] P'$ 

```

```

      moreover assume  $P'RelQ': (\Psi, P', Q') \in X$ 

```

```

      then have  $\Psi \triangleright P' \rightsquigarrow [?X] Q'$  by(rule rSim)

```

```

      moreover from  $\langle eqvt X \rangle P'RelQ'$  have  $eqvt ?X$ 

```

```

        apply(clarsimp simp add: eqvt-def)

```

```

        apply(drule bisimClosed)

```

```

        apply(drule bisimClosed)

```

```

        apply(rule exI)

```

```

        apply(rule conjI)

```

```

        apply assumption

```

```

    apply(rule exI)
    by auto
  ultimately have  $\Psi \triangleright P \rightsquigarrow[?X] Q'$ 
    by(force intro: transitive dest: bisimTransitive)
  moreover assume  $\Psi \triangleright Q' \rightsquigarrow[bisim] Q$ 
  ultimately have  $\Psi \triangleright P \rightsquigarrow[?X] Q$  using  $\langle eqvt ?X \rangle$ 
    by(force intro: transitive dest: bisimTransitive)
}
with  $\langle (\Psi, P, Q) \in ?X \rangle$  show ?case
  by(blast dest: bisimE)
next
case(cExt  $\Psi P Q \Psi'$ )
then show ?case by(blast dest: bisimE intro: rExt)
next
case(cSym  $\Psi P Q$ )
then show ?case
  apply clarsimp
  apply(drule rSym)
  apply clarsimp
  by(metis bisimTransitive bisimE(4))
qed
qed

```

lemma *weakTransitiveCoinduct''*[case-names *cStatEq cSim cExt cSym*, case-conclusion *bisim step*, consumes 2]:

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $X :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

  assumes  $p: (\Psi, P, Q) \in X$ 
  and  $Eqvt: eqvt X$ 
  and  $rStatEq: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies insertAssertion (extractFrame P) \Psi$ 
 $\simeq_F insertAssertion (extractFrame Q) \Psi$ 
  and  $rSim: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow[\{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge$ 
 $\Psi \triangleright P \sim P' \wedge$ 
 $(\Psi, P', Q') \in X \wedge$ 
 $\Psi \triangleright Q' \sim Q\}] Q$ 
  and  $rExt: \bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge$ 
 $(\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\} \implies$ 
 $(\Psi \otimes \Psi', P, Q) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge$ 
 $(\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 
  and  $rSym: \bigwedge \Psi P Q. (\Psi, P, Q) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge$ 
 $(\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\} \implies$ 
 $(\Psi, Q, P) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P',$ 
 $Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 

```

shows $\Psi \triangleright P \sim Q$
 proof –


```

let ?X = {(\Psi, P, Q) | \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q'
\sim Q}
from p have (\Psi, P, Q) \in ?X
  by(blast intro: bisimReflexive)
then show ?thesis
proof(coinduct rule: bisimWeakCoinduct)
  case(cStatEq \Psi P Q)
  then show ?case
    by(blast dest: rStatEq bisimE FrameStatEqTrans)
next
case(cSim \Psi P Q)
{
  fix \Psi P P' Q' Q
  assume \Psi \triangleright P \rightsquigarrow[bisim] P'
  moreover assume P'RelQ': (\Psi, P', Q') \in X
  then have \Psi \triangleright P' \rightsquigarrow[?X] Q' by(rule rSim)
  moreover from \langle eqvt X \rangle P'RelQ' have eqvt ?X
    apply(clarsimp simp add: eqvt-def)
    apply(drule bisimClosed)
    apply(drule bisimClosed)
    apply(rule exI)
    apply(rule conjI)
    apply assumption
    apply(rule exI)
  by auto
  ultimately have \Psi \triangleright P \rightsquigarrow[?X] Q'
    by(force intro: transitive dest: bisimTransitive)
  moreover assume \Psi \triangleright Q' \rightsquigarrow[bisim] Q
  ultimately have \Psi \triangleright P \rightsquigarrow[?X] Q using \langle eqvt ?X \rangle
    by(force intro: transitive dest: bisimTransitive)
}
with \langle (\Psi, P, Q) \in ?X \rangle show ?case
  by(blast dest: bisimE)
next
case(cExt \Psi P Q \Psi')
then show ?case by(rule rExt)
next
case(cSym \Psi P Q)
then show ?case by(rule rSym)
qed
qed

```

lemma *transitiveCoinduct*[case-names *cStatEq cSim cExt cSym*, case-conclusion *bisim step, consumes 2*]:

```

fixes \Psi :: 'b
and P :: ('a, 'b, 'c) psi
and Q :: ('a, 'b, 'c) psi
and X :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set

```

```

assumes  $p: (\Psi, P, Q) \in X$ 
and  $Eqvt: eqvt X$ 
and  $rStatEq: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies insertAssertion (extractFrame R) \Psi' \simeq_F insertAssertion (extractFrame S) \Psi'$ 
and  $rSim: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies \Psi' \triangleright R \rightsquigarrow [(\{\Psi', R, S\} \mid \Psi' R R' S' S. \Psi' \triangleright R \sim R' \wedge ((\Psi', R', S') \in X \vee \Psi' \triangleright R' \sim S'))] S$ 
and  $rExt: \bigwedge \Psi' R S \Psi''. (\Psi', R, S) \in X \implies (\Psi' \otimes \Psi'', R, S) \in X \vee \Psi' \otimes \Psi'' \triangleright R \sim S$ 
and  $rSym: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies (\Psi', S, R) \in X \vee \Psi' \triangleright S \sim R$ 

```

shows $\Psi \triangleright P \sim Q$

proof –

```

from  $p$  have  $(\Psi, P, Q) \in (X \cup bisim)$ 
by  $blast$ 
moreover from  $\langle eqvt X \rangle bisimEqvt$  have  $eqvt (X \cup bisim)$ 
by  $auto$ 
ultimately show  $?thesis$ 
proof( $coinduct\ rule: weakTransitiveCoinduct'$ )
case( $cStatEq \Psi P Q$ )
then show  $?case$ 
by( $blast\ intro: rStatEq\ dest: bisimE$ )
next
case( $cSim \Psi P Q$ )
then show  $?case$ 
apply  $clarsimp$ 
apply ( $erule\ disjE$ )
apply( $blast\ intro: rSim$ )
apply( $drule\ bisimE(2)$ )
apply( $rule\ monotonic, simp$ )
by( $force\ intro: bisimReflexive$ )
next
case( $cExt \Psi P Q \Psi'$ )
then show  $?case$ 
by( $blast\ dest: bisimE\ rExt$ )
next
case( $cSym \Psi P Q$ )
then show  $?case$  by( $blast\ dest: bisimE\ rSym\ intro: bisimReflexive$ )
qed
qed

```

lemma $transitiveCoinduct'$ [$case-names\ cStatEq\ cSim\ cExt\ cSym$, $case-conclusion\ bisim\ step$, $consumes\ 2$]:

```

fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c)\ psi$ 
and  $Q :: ('a, 'b, 'c)\ psi$ 

```

```

and  $X :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$ 

assumes  $p: (\Psi, P, Q) \in X$ 
  and  $\text{Eqvt}: \text{eqvt } X$ 
  and  $r\text{StatEq}: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \text{insertAssertion } (\text{extractFrame } P) \Psi$ 
 $\simeq_F \text{insertAssertion } (\text{extractFrame } Q) \Psi$ 
  and  $r\text{Sim}: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow [(\{\Psi, P, Q\} \mid \Psi P P' Q' Q.$ 
 $\Psi \triangleright P \sim P' \wedge$ 
 $\qquad\qquad\qquad (\Psi, P', Q') \in (X \cup \text{bisim}) \wedge$ 
 $\qquad\qquad\qquad \Psi \triangleright Q' \sim Q)] Q$ 

  and  $r\text{Ext}: \bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in X \implies (\Psi \otimes \Psi', P, Q) \in X \vee \Psi \otimes \Psi' \triangleright P$ 
 $\sim Q$ 
  and  $r\text{Sym}: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies$ 
 $\qquad\qquad\qquad (\Psi, Q, P) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge ((\Psi, P',$ 
 $Q') \in (X \cup \text{bisim})) \wedge \Psi \triangleright Q' \sim Q\}$ 

shows  $\Psi \triangleright P \sim Q$ 
proof –
  from  $p$  have  $(\Psi, P, Q) \in (X \cup \text{bisim})$ 
  by  $\text{blast}$ 
  moreover from  $\langle \text{eqvt } X \rangle \text{ bisimEqvt}$  have  $\text{eqvt } (X \cup \text{bisim})$ 
  by  $\text{auto}$ 
  ultimately show  $?thesis$ 
proof ( $\text{coinduct rule: weakTransitiveCoinduct}'$ )
  case ( $c\text{StatEq } \Psi P Q$ )
  then show  $?case$ 
  by ( $\text{blast intro: rStatEq dest: bisimE}$ )
next
  case ( $c\text{Sim } \Psi P Q$ )
  then show  $?case$ 
  apply –
  apply ( $\text{cases } (\Psi, P, Q) \in X$ )
  apply ( $\text{rule } r\text{Sim}$ )
  apply  $\text{simp}$ 
  apply ( $\text{clarify}$ )
  apply ( $\text{drule } \text{bisimE}(2)$ )
  apply ( $\text{rule } \text{monotonic, simp}$ )
  by ( $\text{force intro: bisimReflexive}$ )
next
  case ( $c\text{Ext } \Psi P Q \Psi'$ )
  then show  $?case$ 
  by ( $\text{blast dest: bisimE rExt}$ )
next
  case ( $c\text{Sym } \Psi P Q$ )
  then show  $?case$ 
  apply  $\text{clarsimp}$ 
  apply ( $\text{erule } \text{disjE}$ )
  apply ( $\text{drule } r\text{Sym}$ )
  apply  $\text{clarsimp}$ 

```

```

    apply(rule exI[where x=Q])
    apply(simp add: bisimReflexive)
    apply(rule exI)
    by(auto intro: bisimReflexive dest: bisimE(4))
  qed
qed

lemma bisimSymmetric:
  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{ psi}$ 

  assumes  $\Psi \triangleright P \sim Q$ 

  shows  $\Psi \triangleright Q \sim P$ 
  using assms
  by(rule bisimE)

lemma eqvTrans[intro]:
  assumes eqv X

  shows eqv  $\{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge ((\Psi, P', Q') \in X \vee \Psi \triangleright P' \sim Q') \wedge \Psi \triangleright Q' \sim Q\}$ 
  using assms
  apply(clarsimp simp add: eqv-def)
  apply(erule disjE)
  using ballE bisimClosed apply fastforce
  by(blast dest: bisimClosed)

lemma eqvWeakTrans[intro]:
  assumes eqv X

  shows eqv  $\{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 
  using assms
  apply(clarsimp simp add: eqv-def)
  using ballE bisimClosed by fastforce

inductive-set
  rel-trancl :: ('b  $\times$  ('a,'b,'c) psi  $\times$  ('a,'b,'c) psi) set  $\Rightarrow$  ('b  $\times$  ('a,'b,'c) psi  $\times$  ('a,'b,'c) psi) set ((-*) [1000] 999)
  for r :: ('b  $\times$  ('a,'b,'c) psi  $\times$  ('a,'b,'c) psi) set
  where
    r-into-rel-trancl [intro, Pure.intro]:  $(\Psi, P, Q) : r \implies (\Psi, P, Q) : r^*$ 
    | rel-trancl-into-rel-trancl [Pure.intro]:  $(\Psi, P, Q) : r^* \implies (\Psi, Q, R) : r \implies (\Psi, P, R) : r^*$ 

lemma rel-trancl-transitive:
  assumes  $(\Psi, P, Q) \in \text{Rel}^*$ 

```

```

    and  $(\Psi, Q, R) \in Rel^*$ 
  shows  $(\Psi, P, R) \in Rel^*$ 
  using  $\langle (\Psi, Q, R) \in Rel^* \rangle \langle (\Psi, P, Q) \in Rel^* \rangle$ 
  by(induct rule: rel-trancl.induct) (auto intro: rel-trancl-into-rel-trancl)

lemma rel-trancl-eqvt:
  assumes eqvt X
  shows eqvt(X*)
proof -
  {
    fix p::name prm
    and  $\Psi P Q$ 
    assume  $(\Psi, P, Q) \in X^*$ 
    then have  $(p \cdot (\Psi, P, Q)) \in X^*$ 
    proof(induct rule: rel-trancl.induct)
      case(r-into-rel-trancl  $\Psi P Q$ )
      with  $\langle eqvt X \rangle$  have  $(p \cdot (\Psi, P, Q)) \in X$ 
      unfolding eqvt-def by auto
      then show ?case by auto
    next
      case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
      from  $\langle (\Psi, Q, R) \in X \rangle \langle eqvt X \rangle$  have  $(p \cdot (\Psi, Q, R)) \in X$ 
      unfolding eqvt-def by auto
      then have  $(p \cdot (\Psi, Q, R)) \in X^*$ 
      by auto
      with  $\langle p \cdot (\Psi, P, Q) \in X^* \rangle$  show ?case by(auto dest: rel-trancl-transitive)
    qed
  }
  then show ?thesis unfolding eqvt-def
  by auto
qed

lemma bisimStarWeakCoinduct[consumes 2, case-names cStatEq cSim cExt cSym]:
  fixes F :: 'b
  and P :: ('a, 'b, 'c) psi
  and Q :: ('a, 'b, 'c) psi
  and X :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

  assumes  $(\Psi, P, Q) \in X$ 
  and eqvt X
  and rStatEq:  $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies insertAssertion (extractFrame R) \Psi' \simeq_F insertAssertion (extractFrame S) \Psi'$ 
  and rSim:  $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies \Psi' \triangleright R \rightsquigarrow[X^*] S$ 
  and rExt:  $\bigwedge \Psi' R S \Psi''. (\Psi', R, S) \in X \implies (\Psi' \otimes \Psi'', R, S) \in X$ 
  and rSym:  $\bigwedge \Psi' R S. (\Psi', R, S) \in X \implies (\Psi', S, R) \in X$ 

  shows  $(\Psi, P, Q) \in bisim$ 
proof -
  have eqvt(X*) using  $\langle eqvt X \rangle$ 

```

```

  by(rule rel-trancl-eqvt)
from  $\langle (\Psi, P, Q) \in X \rangle$ 
have  $(\Psi, P, Q) \in X^*$ 
  by force
then show ?thesis
proof(coinduct rule: bisimWeakCoinduct)
  case(cSim  $\Psi' R S$ )
  then show ?case
  proof(induct rule: rel-trancl.induct)
    case(r-into-rel-trancl  $\Psi P Q$ )
    then show ?case
    by(rule rSim)
  next
  case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
  note  $\langle \Psi \triangleright P \rightsquigarrow[X^*] Q \rangle$ 
  moreover from  $\langle (\Psi, Q, R) \in X \rangle$  have  $\Psi \triangleright Q \rightsquigarrow[X^*] R$ 
    by(rule rSim)
  moreover note  $\langle eqvt(X^*) \rangle$ 
  moreover have  $\{(\Psi, P, R) \mid \Psi P R. \exists Q. (\Psi, P, Q) \in X^* \wedge (\Psi, Q, R) \in X^*\} \subseteq X^*$ 
    by(blast intro: rel-trancl-transitive)
  ultimately show ?case
    using transitive by meson
  qed
next
case (cStatEq  $\Psi P Q$ )
then show ?case
proof(induct rule: rel-trancl.induct)
  case(r-into-rel-trancl  $\Psi P Q$ )
  then show ?case
  by(rule rStatEq)
next
case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
from  $\langle (\Psi, Q, R) \in X \rangle$  have insertAssertion (extractFrame Q)  $\Psi \simeq_F insertAssertion (extractFrame R) \Psi$ 
  by(rule rStatEq)
with  $\langle insertAssertion (extractFrame P) \Psi \simeq_F insertAssertion (extractFrame Q) \Psi \rangle$ 
show ?case
  by(rule FrameStatEqTrans)
qed
next
case(cExt  $\Psi P Q \Psi'$ )
then show ?case
proof(induct rule: rel-trancl.induct)
  case(r-into-rel-trancl  $\Psi P Q$ )
  then have  $(\Psi \otimes \Psi', P, Q) \in X$  by(rule rExt)
  then show ?case
  by force

```

```

next
  case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
  from  $\langle (\Psi, Q, R) \in X \rangle$  have  $(\Psi \otimes \Psi', Q, R) \in X$  by(rule rExt)
  then have  $(\Psi \otimes \Psi', Q, R) \in X^*$  by force
  with  $\langle (\Psi \otimes \Psi', P, Q) \in X^* \rangle$ 
  show ?case
    by(rule rel-trancl-transitive)
qed
next
case(cSym  $\Psi P Q$ )
then show ?case
proof(induct rule: rel-trancl.induct)
  case(r-into-rel-trancl  $\Psi P Q$ )
  then have  $(\Psi, Q, P) \in X$  by(rule rSym)
  then show ?case
    by force
next
case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
from  $\langle (\Psi, Q, R) \in X \rangle$  have  $(\Psi, R, Q) \in X$  by(rule rSym)
then have  $(\Psi, R, Q) \in X^*$  by force
then show ?case using  $\langle (\Psi, Q, P) \in X^* \rangle$ 
  by(rule rel-trancl-transitive)
qed
qed
qed

```

lemma *weakTransitiveStarCoinduct*[*case-names cStatEq cSim cExt cSym, case-conclusion bisim step, consumes 2*]:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $X :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

assumes  $p: (\Psi, P, Q) \in X$ 
  and  $Eqvt: eqvt X$ 
  and  $rStatEq: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies insertAssertion (extractFrame P) \Psi \simeq_F insertAssertion (extractFrame Q) \Psi$ 
  and  $rSim: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies \Psi \triangleright P \rightsquigarrow [(\{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\})^*] Q$ 
  and  $rExt: \bigwedge \Psi P Q \Psi'. (\Psi, P, Q) \in X \implies (\Psi \otimes \Psi', P, Q) \in X$ 
  and  $rSym: \bigwedge \Psi P Q. (\Psi, P, Q) \in X \implies (\Psi, Q, P) \in X$ 

```

shows $\Psi \triangleright P \sim Q$

proof –

```

let ?X =  $\{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 
from  $p$  have  $(\Psi, P, Q) \in ?X$ 

```

```

    by(blast intro: bisimReflexive)
  moreover from  $\langle \text{eqvt } X \rangle$  have eqvt ?X by auto
  ultimately show ?thesis
  proof(coinduct rule: bisimStarWeakCoinduct)
    case(cStatEq  $\Psi P Q$ )
    then show ?case
      by(blast dest: rStatEq bisimE FrameStatEqTrans)
  next
  case(cSim  $\Psi P Q$ )
  then obtain  $P' Q'$  where  $\Psi \triangleright P \sim P'$  and  $(\Psi, P', Q') \in X$  and  $\Psi \triangleright Q' \sim$ 
Q
    by blast
  then have  $\Psi \triangleright P \rightsquigarrow[\textit{bisim}] P'$  and  $\Psi \triangleright Q' \rightsquigarrow[\textit{bisim}] Q$ 
    by(auto dest: bisimE)
  {
    fix  $\Psi P Q$ 
    and  $Q'::('a, 'b, 'c)$  psi
    assume  $\Psi \triangleright P \sim Q$ 
    and  $(\Psi, Q, Q') \in ?X^*$ 
    note  $\langle (\Psi, Q, Q') \in ?X^* \rangle$   $\langle \Psi \triangleright P \sim Q \rangle$ 
    then have  $(\Psi, P, Q') \in ?X^*$ 
    proof(induct rule: rel-trancl.inducts)
      case(r-into-rel-trancl  $\Psi Q Q'$ ) then show ?case
        by(blast dest: bisimTransitive)
    next
      case(rel-trancl-into-rel-trancl  $\Psi P' Q Q'$ )
      then show ?case
        by(blast dest: rel-trancl-transitive)
    qed
  }
  note tonic = this
  {
    fix  $\Psi P Q$ 
    and  $Q'::('a, 'b, 'c)$  psi
    assume  $(\Psi, P, Q) \in ?X^*$ 
    and  $\Psi \triangleright Q \sim Q'$ 
    then have  $(\Psi, P, Q') \in ?X^*$ 
    proof(induct arbitrary:  $Q'$  rule: rel-trancl.inducts)
      case(r-into-rel-trancl  $\Psi P Q$ ) then show ?case
        by(blast dest: bisimTransitive)
    next
      case(rel-trancl-into-rel-trancl  $\Psi P P' Q$ )
      from  $\langle (\Psi, P', Q) \in ?X \rangle$   $\langle \Psi \triangleright Q \sim Q' \rangle$  have  $(\Psi, P', Q') \in ?X$ 
        by(blast dest: bisimTransitive)
      then have  $(\Psi, P', Q') \in ?X^*$ 
        by blast
      with  $\langle (\Psi, P, P') \in ?X^* \rangle$ 
      show ?case
        by(rule rel-trancl-transitive)
  }

```



```

    qed
  }
  note tonic2 = this
  from ⟨(Ψ, P', Q') ∈ X⟩ have Ψ ▷ P'  $\rightsquigarrow$ [?X*] Q'
    by(rule rSim)
  with ⟨Ψ ▷ P  $\rightsquigarrow$ [bisim] P'⟩ have Ψ ▷ P  $\rightsquigarrow$ [?X*] Q'
    apply –
    apply(rule transitive)
      apply assumption
      apply assumption
      apply(rule rel-trancl-eqt)
      apply(rule ⟨eqvt ?X⟩)
    by(blast intro: tonic)
  with ⟨Ψ ▷ Q'  $\rightsquigarrow$ [bisim] Q⟩ show ?case
    apply –
    apply(rule transitive)
      apply assumption
      apply assumption
      apply(rule rel-trancl-eqt)
      apply(rule ⟨eqvt ?X⟩)
    by(blast intro: tonic2)
  next
  case(cExt Ψ P Q Ψ')
  then show ?case by(blast dest: bisimE intro: rExt)
  next
  case(cSym Ψ P Q)
  then show ?case by(blast dest: bisimE intro: rSym)
  qed
qed

```

lemma *weakTransitiveStarCoinduct* [case-names cStatEq cSim cExt cSym, case-conclusion bisim step, consumes 2]:

```

  fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi
  and Q :: ('a, 'b, 'c) psi
  and X :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

  assumes p: (Ψ, P, Q) ∈ X
  and Eqvt: eqvt X
  and rStatEq:  $\bigwedge$  Ψ P Q. (Ψ, P, Q) ∈ X  $\implies$  insertAssertion (extractFrame P) Ψ
 $\simeq_F$  insertAssertion (extractFrame Q) Ψ
  and rSim:  $\bigwedge$  Ψ P Q. (Ψ, P, Q) ∈ X  $\implies$  Ψ ▷ P  $\rightsquigarrow$ [({(Ψ, P, Q) | Ψ P P' Q' Q.
Ψ ▷ P  $\sim$  P'  $\wedge$ 

$$\begin{array}{l} (\Psi, P', Q') \in X \wedge \\ \Psi \triangleright Q' \sim Q \}]^* Q \end{array}$$

  and rExt:  $\bigwedge$  Ψ P Q Ψ'. (Ψ, P, Q) ∈ X  $\implies$  (Ψ ⊗ Ψ', P, Q) ∈ X
  and rSym:  $\bigwedge$  Ψ P Q. (Ψ, P, Q) ∈ X  $\implies$ 

$$(\Psi, Q, P) \in \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$$


```

```

shows  $\Psi \triangleright P \sim Q$ 
proof -
  let  $?X = \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \triangleright Q' \sim Q\}$ 
  from  $p$  have  $(\Psi, P, Q) \in ?X$ 
  by(blast intro: bisimReflexive)
  moreover from  $\langle \text{eqvt } X \rangle$  have eqvt  $?X$  by auto
  ultimately show ?thesis
  proof(coinduct rule: bisimStarWeakCoinduct)
    case(cStatEq  $\Psi P Q$ )
    then show ?case
      by(blast dest: rStatEq bisimE FrameStatEqTrans)
  next
    case(cSim  $\Psi P Q$ )
    then obtain  $P' Q'$  where  $\Psi \triangleright P \sim P'$  and  $(\Psi, P', Q') \in X$  and  $\Psi \triangleright Q' \sim$ 
 $Q$ 
      by blast
    then have  $\Psi \triangleright P \rightsquigarrow[\text{bisim}] P'$  and  $\Psi \triangleright Q' \rightsquigarrow[\text{bisim}] Q$ 
    by(auto dest: bisimE)
    {
      fix  $\Psi P Q$ 
      and  $Q'::('a, 'b, 'c)$  psi
      assume  $\Psi \triangleright P \sim Q$ 
      and  $(\Psi, Q, Q') \in ?X^*$ 
      note  $\langle (\Psi, Q, Q') \in ?X^* \rangle \langle \Psi \triangleright P \sim Q \rangle$ 
      then have  $(\Psi, P, Q') \in ?X^*$ 
      proof(induct rule: rel-trancl.inducts)
        case(r-into-rel-trancl  $\Psi Q Q'$ ) then show ?case
          by(blast dest: bisimTransitive)
      next
        case(rel-trancl-into-rel-trancl  $\Psi P' Q Q'$ )
        then show ?case
          by(blast dest: rel-trancl-transitive)
      qed
    }
  }
  note tonic = this
  {
    fix  $\Psi P Q$ 
    and  $Q'::('a, 'b, 'c)$  psi
    assume  $(\Psi, P, Q) \in ?X^*$ 
    and  $\Psi \triangleright Q \sim Q'$ 
    then have  $(\Psi, P, Q') \in ?X^*$ 
    proof(induct arbitrary: Q' rule: rel-trancl.inducts)
      case(r-into-rel-trancl  $\Psi P Q$ ) then show ?case
        by(blast dest: bisimTransitive)
    next
      case(rel-trancl-into-rel-trancl  $\Psi P P' Q$ )
      from  $\langle (\Psi, P', Q) \in ?X \rangle \langle \Psi \triangleright Q \sim Q' \rangle$  have  $(\Psi, P', Q') \in ?X$ 

```

```

      by(blast dest: bisimTransitive)
    then have  $(\Psi, P', Q') \in ?X^*$ 
      by blast
    with  $\langle \Psi, P, P' \rangle \in ?X^*$ 
    show ?case
      by(rule rel-trancl-transitive)
  qed
}
note tonic2 = this
from  $\langle \Psi, P', Q' \rangle \in X$  have  $\Psi \triangleright P' \rightsquigarrow[?X^*] Q'$ 
  by(rule rSim)
with  $\langle \Psi \triangleright P \rightsquigarrow[bisim] P' \rangle$  have  $\Psi \triangleright P \rightsquigarrow[?X^*] Q'$ 
  apply -
  apply(rule transitive)
  apply assumption
  apply assumption
  apply(rule rel-trancl-eqvt)
  apply(rule eqvt ?X)
  by(blast intro: tonic)
with  $\langle \Psi \triangleright Q' \rightsquigarrow[bisim] Q \rangle$  show ?case
  apply -
  apply(rule transitive)
  apply assumption
  apply assumption
  apply(rule rel-trancl-eqvt)
  apply(rule eqvt ?X)
  by(blast intro: tonic2)
next
case(cExt  $\Psi P Q \Psi'$ )
then show ?case by(blast dest: bisimE intro: rExt)
next
case(cSym  $\Psi P Q$ )
then show ?case
  apply clarsimp
  apply(drule rSym)
  apply clarsimp
  by(metis bisimTransitive bisimE(4))
qed
qed

```

lemma *transitiveStarCoinduct*[*case-names cStatEq cSim cExt cSym, case-conclusion bisim step, consumes 2*]:

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $X :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

```

```

assumes  $p: (\Psi, P, Q) \in X$ 
and Eqvt: eqvt X

```

and $rStatEq: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies insertAssertion (extractFrame R) \Psi' \simeq_F insertAssertion (extractFrame S) \Psi'$
and $rSim: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies \Psi' \triangleright R \rightsquigarrow [(\{\Psi', R, S\} \mid \Psi' R R' S' S. \Psi' \triangleright R \sim R' \wedge$
 $((\Psi', R', S') \in X \vee \Psi' \triangleright$
 $R' \sim S') \wedge$
 $\Psi' \triangleright S' \sim S\})*] S$
and $rExt: \bigwedge \Psi' R S \Psi''. (\Psi', R, S) \in X \implies (\Psi' \otimes \Psi'', R, S) \in X \vee \Psi' \otimes \Psi'' \triangleright R \sim S$
and $rSym: \bigwedge \Psi' R S. (\Psi', R, S) \in X \implies (\Psi', S, R) \in X \vee \Psi' \triangleright S \sim R$

shows $\Psi \triangleright P \sim Q$

proof –

from p **have** $(\Psi, P, Q) \in (X \cup bisim)$

by $blast$

moreover from $\langle eqvt X \rangle bisimEqvt$ **have** $eqvt (X \cup bisim)$

by $auto$

ultimately show $?thesis$

proof($coinduct$ rule: $weakTransitiveStarCoinduct'$)

case($cStatEq \Psi P Q$)

then show $?case$

by($blast$ intro: $rStatEq$ dest: $bisimE$)

next

case($cSim \Psi P Q$)

then show $?case$

apply $clarsimp$

apply ($erule$ $disjE$)

apply($blast$ intro: $rSim$)

apply($drule$ $bisimE(2)$)

apply($rule$ $monotonic, simp$)

by($force$ intro: $bisimReflexive$)

next

case($cExt \Psi P Q \Psi'$)

then show $?case$

by($blast$ dest: $bisimE$ $rExt$)

next

case($cSym \Psi P Q$)

then show $?case$ **by**($blast$ dest: $bisimE$ $rSym$ intro: $bisimReflexive$)

qed

qed

end

end

theory $Sim-Pres$

imports $Simulation$

begin

This file is a (heavily modified) variant of the theory $Psi_Calculi.Sim_Pres$

from [1].

context *env* **begin**

lemma *inputPres*:

```

fixes  $\Psi$     :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $Rel$    :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $xvec$   :: name list
and  $N$      :: 'a

```

assumes $PRelQ$: $\bigwedge Tvec. length\ xvec = length\ Tvec \implies (\Psi, P[xvec ::= Tvec], Q[xvec ::= Tvec]) \in Rel$

shows $\Psi \triangleright M(\lambda*xvec\ N).P \rightsquigarrow[Rel] M(\lambda*xvec\ N).Q$

proof –

```

{
  fix  $\alpha\ Q'$ 
  assume  $\Psi \triangleright M(\lambda*xvec\ N).Q \mapsto_{\alpha} < Q'$ 
  then have  $\exists P'. \Psi \triangleright M(\lambda*xvec\ N).P \mapsto_{\alpha} < P' \wedge (\Psi, P', Q') \in Rel$ 
    by(induct rule: inputCases) (auto intro: Input BrInput PRelQ)
}
then show ?thesis
by(auto simp add: simulation-def residual.inject psi.inject)

```

qed

lemma *outputPres*:

```

fixes  $\Psi$     :: 'b
and  $P$      :: ('a, 'b, 'c) psi
and  $Rel$    :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
and  $Q$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $N$      :: 'a

```

assumes $PRelQ$: $(\Psi, P, Q) \in Rel$

shows $\Psi \triangleright M\langle N \rangle.P \rightsquigarrow[Rel] M\langle N \rangle.Q$

proof –

```

{
  fix  $\alpha\ Q'$ 
  assume  $\Psi \triangleright M\langle N \rangle.Q \mapsto_{\alpha} < Q'$ 
  then have  $\exists P'. \Psi \triangleright M\langle N \rangle.P \mapsto_{\alpha} < P' \wedge (\Psi, P', Q') \in Rel$ 
    by(induct rule: outputCases) (auto intro: Output BrOutput PRelQ)
}
then show ?thesis
by(auto simp add: simulation-def residual.inject psi.inject)

```

qed

lemma *casePres*:

fixes Ψ :: 'b
and CsP :: ('c × ('a, 'b, 'c) psi) list
and Rel :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set
and CsQ :: ('c × ('a, 'b, 'c) psi) list
and M :: 'a
and N :: 'a

assumes $PRelQ$: $\bigwedge \varphi Q. (\varphi, Q) \in set\ CsQ \implies \exists P. (\varphi, P) \in set\ CsP \wedge guarded\ P \wedge (\Psi, P, Q) \in Rel$
and Sim : $\bigwedge \Psi' R S. (\Psi', R, S) \in Rel \implies \Psi' \triangleright R \rightsquigarrow[Rel] S$
and $Rel \subseteq Rel'$

shows $\Psi \triangleright Cases\ CsP \rightsquigarrow[Rel'] Cases\ CsQ$

proof –

{
fix $\alpha Q'$
assume $\Psi \triangleright Cases\ CsQ \mapsto_{\alpha} \prec Q'$ **and** $bn\ \alpha\ \#\#*\ CsP$ **and** $bn\ \alpha\ \#\#\ \Psi$
then have $\exists P'. \Psi \triangleright Cases\ CsP \mapsto_{\alpha} \prec P' \wedge (\Psi, P', Q') \in Rel'$
proof(*induct rule: caseCases*)
case(*cCase* φQ)
from $\langle (\varphi, Q) \in set\ CsQ \rangle$ **obtain** P **where** $(\varphi, P) \in set\ CsP$ **and** *guarded* P
and $(\Psi, P, Q) \in Rel$
by(*metis* $PRelQ$)
from $\langle (\Psi, P, Q) \in Rel \rangle$ **have** $\Psi \triangleright P \rightsquigarrow[Rel] Q$
by(*rule* Sim)
moreover from $\langle bn\ \alpha\ \#\#\ CsP \rangle \langle (\varphi, P) \in set\ CsP \rangle$ **have** $bn\ \alpha\ \#\#\ P$
by(*auto dest: memFreshChain*)
moreover note $\langle \Psi \triangleright Q \mapsto_{\alpha} \prec Q' \rangle \langle bn\ \alpha\ \#\#\ \Psi \rangle$
ultimately obtain P' **where** $PTrans: \Psi \triangleright P \mapsto_{\alpha} \prec P'$ **and** $P'RelQ': (\Psi, P', Q') \in Rel$
by(*blast dest: simE*)
from $PTrans \langle (\varphi, P) \in set\ CsP \rangle \langle \Psi \vdash \varphi \rangle \langle guarded\ P \rangle$ **have** $\Psi \triangleright Cases\ CsP \mapsto_{\alpha} \prec P'$
by(*rule* $Case$)
moreover from $P'RelQ' \langle Rel \subseteq Rel' \rangle$ **have** $(\Psi, P', Q') \in Rel'$
by *blast*
ultimately show *?case*
by *blast*
qed
}
then show *?thesis*
by(*auto simp add: simulation-def residual.inject psi.inject*)
qed

lemma *resPres*:

fixes Ψ :: 'b
and P :: ('a, 'b, 'c) psi
and Rel :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

```

and  $Q$   :: ('a, 'b, 'c) psi
and  $x$    :: name
and  $Rel'$  :: ('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set

assumes  $PSimQ$ :  $\Psi \triangleright P \rightsquigarrow[Rel] Q$ 
and  $eqvt\ Rel'$ 
and  $x \# \Psi$ 
and  $Rel \subseteq Rel'$ 
and  $C1$ :  $\bigwedge \Psi' R S\ yvec. \llbracket (\Psi', R, S) \in Rel; yvec \#* \Psi \rrbracket \implies (\Psi', (\nu^* yvec) R, (\nu^* yvec) S) \in Rel'$ 

shows  $\Psi \triangleright (\nu x) P \rightsquigarrow[Rel'] (\nu x) Q$ 
proof -
  note  $\langle eqvt\ Rel' \rangle \langle x \# \Psi \rangle$ 
  moreover have  $x \# (\nu x) P$  and  $x \# (\nu x) Q$  by ( $simp\ add: abs-fresh$ ) +
  ultimately show ?thesis
  proof ( $induct\ rule: simIFresh[where\ C=(\ )]$ )
    case ( $cSim\ \alpha\ Q'$ )
      from  $\langle bn\ \alpha \#* (\nu x) P \rangle \langle bn\ \alpha \#* (\nu x) Q \rangle \langle x \# \alpha \rangle$  have  $bn\ \alpha \#* P$  and  $bn\ \alpha \#* Q$  by  $simp+$ 
      from  $\langle \Psi \triangleright (\nu x) Q \mapsto \alpha \prec Q' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# Q' \rangle \langle bn\ \alpha \#* \Psi \rangle \langle bn\ \alpha \#* Q \rangle \langle bn\ \alpha \#* subject\ \alpha \rangle$ 
         $\langle bn\ \alpha \#* \Psi \rangle \langle bn\ \alpha \#* P \rangle \langle x \# \alpha \rangle$ 
      show ?case
      proof ( $induct\ rule: resCases[where\ C=P]$ )
        case ( $cOpen\ M\ xvec1\ xvec2\ y\ N\ Q'$ )
          from  $\langle bn\ (M(\nu^*(xvec1 @ y \# xvec2))) \langle N \rangle \#* \Psi \rangle$  have  $xvec1 \#* \Psi$  and  $y \# \Psi$ 
        and  $xvec2 \#* \Psi$  by  $simp+$ 
          from  $\langle bn\ (M(\nu^*(xvec1 @ y \# xvec2))) \langle N \rangle \#* P \rangle$  have  $xvec1 \#* P$  and  $y \# P$ 
        and  $xvec2 \#* P$  by  $simp+$ 
          from  $\langle x \# (M(\nu^*(xvec1 @ y \# xvec2))) \langle N \rangle \rangle$  have  $x \# xvec1$  and  $x \neq y$  and  $x \# xvec2$  and  $x \# M$  by  $simp+$ 
          from  $PSimQ \langle \Psi \triangleright Q \mapsto M(\nu^*(xvec1 @ xvec2)) \langle ((x, y) \cdot N) \rangle \prec ((x, y) \cdot Q') \rangle$ 
             $\langle xvec1 \#* \Psi \rangle \langle xvec2 \#* \Psi \rangle \langle xvec1 \#* P \rangle \langle xvec2 \#* P \rangle$ 
          obtain  $P'$  where  $PTrans: \Psi \triangleright P \mapsto M(\nu^*(xvec1 @ xvec2)) \langle ((x, y) \cdot N) \rangle \prec P'$  and  $P'RelQ': (\Psi, P', ((x, y) \cdot Q')) \in Rel$ 
          by ( $force\ dest: simE$ )
          from  $\langle y \in supp\ N \rangle \langle x \neq y \rangle$  have  $x \in supp\ ((x, y) \cdot N)$ 
          apply -
          apply ( $drule\ pt-set-bij2[OF\ pt-name-inst, OF\ at-name-inst, where\ pi=((x, y))]$ )
          by ( $simp\ add: eqvts\ calc-atm$ )
          with  $PTrans \langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec1 \rangle \langle x \# xvec2 \rangle$ 
          have  $\Psi \triangleright (\nu x) P \mapsto M(\nu^*(xvec1 @ x \# xvec2)) \langle ((x, y) \cdot N) \rangle \prec P'$ 
          by ( $metis\ Open$ )
          then have  $((x, y) \cdot \Psi) \triangleright ((x, y) \cdot (\nu x) P) \mapsto ((x, y) \cdot (M(\nu^*(xvec1 @ x \# xvec2)) \langle ((x, y) \cdot N) \rangle \prec P'))$ 
          by ( $rule\ eqvts$ )

```

with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle y \# P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle \langle y \# xvec1 \rangle \langle x \# xvec2 \rangle \langle y \# xvec2 \rangle \langle x \neq y \rangle$
have $\Psi \triangleright (\nu x)P \mapsto M(\nu*(xvec1@y\#xvec2))\langle N \rangle \prec ([x, y]) \cdot P'$ **by** (*simp add: eqvts calc-atm alphaRes*)
moreover from $P'RelQ' \langle Rel \subseteq Rel' \rangle \langle eqvt Rel' \rangle$ **have** $([(y, x)] \cdot \Psi, [(y, x)] \cdot P', [(y, x)] \cdot [(x, y)] \cdot Q') \in Rel'$
by (*force simp add: eqvt-def*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **have** $(\Psi, [(x, y)] \cdot P', Q') \in Rel'$ **by** (*simp add: name-swap*)
ultimately show ?*case by blast*
next
case (*cBrOpen M xvec1 xvec2 y N Q'*)
from $\langle bn \ (iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* \Psi \rangle$ **have** $xvec1 \#* \Psi$ **and** $y \# \Psi$
and $xvec2 \#* \Psi$ **by** *simp+*
from $\langle bn \ (iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* P \rangle$ **have** $xvec1 \#* P$ **and** $y \# P$
and $xvec2 \#* P$ **by** *simp+*
from $\langle x \# (iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \rangle$ **have** $x \# xvec1$ **and** $x \neq y$ **and** $x \# xvec2$ **and** $x \# M$ **by** *simp+*
from $PSimQ \langle \Psi \triangleright Q \mapsto iM(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N \rangle \prec ([x, y]) \cdot Q' \rangle$
 $\langle xvec1 \#* \Psi \rangle \langle xvec2 \#* \Psi \rangle \langle xvec1 \#* P \rangle \langle xvec2 \#* P \rangle$
obtain P' **where** $PTrans: \Psi \triangleright P \mapsto iM(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N \rangle \prec P'$ **and** $P'RelQ': (\Psi, P', [(x, y)] \cdot Q') \in Rel$
by (*force dest: simE*)
from $\langle y \in supp N \rangle \langle x \neq y \rangle$ **have** $x \in supp([x, y]) \cdot N$
apply –
apply (*drule pt-set-bij2[OF pt-name-inst, OF at-name-inst, where pi=[(x, y)]]*)
by (*simp add: eqvts calc-atm*)
with $PTrans \langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec1 \rangle \langle x \# xvec2 \rangle$
have $\Psi \triangleright (\nu x)P \mapsto iM(\nu*(xvec1@x\#xvec2))\langle [(x, y)] \cdot N \rangle \prec P'$
by (*metis BrOpen*)
then have $([(x, y)] \cdot \Psi) \triangleright ([x, y]) \cdot (\nu x)P \mapsto ([x, y]) \cdot (iM(\nu*(xvec1@x\#xvec2))\langle [(x, y)] \cdot N \rangle) \prec P'$
by (*rule eqvts*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle y \# P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle \langle y \# xvec1 \rangle \langle x \# xvec2 \rangle \langle y \# xvec2 \rangle \langle x \neq y \rangle$
have $\Psi \triangleright (\nu x)P \mapsto iM(\nu*(xvec1@y\#xvec2))\langle N \rangle \prec ([x, y]) \cdot P'$ **by** (*simp add: eqvts calc-atm alphaRes*)
moreover from $P'RelQ' \langle Rel \subseteq Rel' \rangle \langle eqvt Rel' \rangle$ **have** $([(y, x)] \cdot \Psi, [(y, x)] \cdot P', [(y, x)] \cdot [(x, y)] \cdot Q') \in Rel'$
by (*force simp add: eqvt-def*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ **have** $(\Psi, [(x, y)] \cdot P', Q') \in Rel'$ **by** (*simp add: name-swap*)
ultimately show ?*case by blast*
next
case (*cRes Q'*)
from $PSimQ \langle \Psi \triangleright Q \mapsto \alpha \prec Q' \rangle \langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* P \rangle$
obtain P' **where** $PTrans: \Psi \triangleright P \mapsto \alpha \prec P'$ **and** $P'RelQ': (\Psi, P', Q') \in$

Rel

```

  by(blast dest: simE)
  from PTrans  $\langle x \# \Psi \rangle \langle x \# \alpha \rangle$  have  $\Psi \triangleright (\nu x)P \mapsto \alpha \prec (\nu x)P'$ 
  by(rule Scope)
  moreover from P'RelQ'  $\langle x \# \Psi \rangle$  have  $(\Psi, (\nu*[x])P', (\nu*[x])Q') \in Rel'$ 
  by(intro C1[where yvec=[x]] simp+)
  moreover then have  $(\Psi, (\nu x)P', (\nu x)Q') \in Rel'$ 
  by (metis resChain.base resChain.step)
  ultimately show ?case by blast
next
case(cBrClose M xvec N Q')
from PSimQ  $\langle \Psi \triangleright Q \mapsto \imath M(\nu*xvec)\langle N \rangle \prec Q' \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$ 
  obtain P' where PTrans:  $\Psi \triangleright P \mapsto \imath M(\nu*xvec)\langle N \rangle \prec P'$  and P'RelQ':
( $\Psi, P', Q') \in Rel$ 
  by(force dest: simE)
  with  $\langle x \# \Psi \rangle \langle xvec \#* \Psi \rangle$ 
  have  $(x\#xvec) \#* \Psi$  by simp

  from P'RelQ'
  have  $(\Psi, (\nu*(x\#xvec))P', (\nu*(x\#xvec))Q') \in Rel'$ 
  using  $\langle (x\#xvec) \#* \Psi \rangle$ 
  by(rule C1)
  then have  $(\Psi, (\nu x)((\nu*xvec)P'), (\nu x)((\nu*xvec)Q')) \in Rel'$ 
  by simp

  moreover from  $\langle \Psi \triangleright P \mapsto \imath M(\nu*xvec)\langle N \rangle \prec P' \rangle \langle x \in \text{supp } M \rangle \langle x \# \Psi \rangle$ 
  have  $\Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)((\nu*xvec)P')$ 
  by(rule BrClose)
  ultimately show ?case
  by blast
qed
qed
qed

lemma resChainPres:
  fixes  $\Psi$     :: 'b
  and P      :: ('a, 'b, 'c) psi
  and Rel    :: ('b  $\times$  ('a, 'b, 'c) psi  $\times$  ('a, 'b, 'c) psi) set
  and Q      :: ('a, 'b, 'c) psi
  and xvec   :: name list

assumes PSimQ:  $\Psi \triangleright P \rightsquigarrow[Rel] Q$ 
  and eqt Rel
  and xvec  $\#* \Psi$ 
  and C1:  $\bigwedge \Psi' R S \text{ yvec. } \llbracket (\Psi', R, S) \in Rel; \text{ yvec} \#* \Psi \rrbracket \implies (\Psi', (\nu*\text{yvec})R, (\nu*\text{yvec})S) \in Rel$ 

shows  $\Psi \triangleright (\nu*xvec)P \rightsquigarrow[Rel] (\nu*xvec)Q$ 
  using  $\langle xvec \#* \Psi \rangle$ 

```

```

proof(induct xvec)
  case Nil
  from PSimQ show ?case by simp
next
  case(Cons x xvec)
  from  $\langle x \# xvec \rangle \#* \Psi$  have  $x \# \Psi$  and  $xvec \#* \Psi$  by simp+
  from  $\langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu * xvec)P \rightsquigarrow[Rel] (\nu * xvec)Q$  by(rule Cons)
  moreover note  $\langle eqvt Rel \rangle \langle x \# \Psi \rangle$ 
  moreover have  $Rel \subseteq Rel$  by simp
  ultimately have  $\Psi \triangleright (\nu x)((\nu * xvec)P) \rightsquigarrow[Rel] (\nu x)((\nu * xvec)Q)$  using C1
  by(rule resPres)
  then show ?case by simp
qed

lemma parPres:
  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $R :: ('a, 'b, 'c) psi$ 
  and  $Rel' :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

assumes PRelQ:  $\bigwedge A_R \Psi_R. \llbracket extractFrame R = \langle A_R, \Psi_R \rangle; A_R \#* \Psi; A_R \#* P; A_R \#* Q \rrbracket \implies (\Psi \otimes \Psi_R, P, Q) \in Rel$ 
  and Eqvt: eqvt Rel
  and Eqvt': eqvt Rel'

and StatImp:  $\bigwedge \Psi' S T. (\Psi', S, T) \in Rel \implies insertAssertion (extractFrame T) \Psi' \hookrightarrow_F insertAssertion (extractFrame S) \Psi'$ 
and Sim:  $\bigwedge \Psi' S T. (\Psi', S, T) \in Rel \implies \Psi' \triangleright S \rightsquigarrow[Rel] T$ 
and Ext:  $\bigwedge \Psi' S T \Psi''. \llbracket (\Psi', S, T) \in Rel \rrbracket \implies (\Psi' \otimes \Psi'', S, T) \in Rel$ 

and C1:  $\bigwedge \Psi' S T A_U \Psi_U U. \llbracket (\Psi' \otimes \Psi_U, S, T) \in Rel; extractFrame U = \langle A_U, \Psi_U \rangle; A_U \#* \Psi'; A_U \#* S; A_U \#* T \rrbracket \implies (\Psi', S \parallel U, T \parallel U) \in Rel'$ 
and C2:  $\bigwedge \Psi' S T xvec. \llbracket (\Psi', S, T) \in Rel'; xvec \#* \Psi \rrbracket \implies (\Psi', (\nu * xvec)S, (\nu * xvec)T) \in Rel'$ 
and C3:  $\bigwedge \Psi' S T \Psi''. \llbracket (\Psi', S, T) \in Rel; \Psi' \simeq \Psi'' \rrbracket \implies (\Psi'', S, T) \in Rel$ 

shows  $\Psi \triangleright P \parallel R \rightsquigarrow[Rel'] Q \parallel R$ 
  using Eqvt'
proof(induct rule: simI[of - - - ()])
  case(cSim  $\alpha QR$ )
  from  $\langle bn \alpha \#* (P \parallel R) \rangle \langle bn \alpha \#* (Q \parallel R) \rangle$ 
  have  $bn \alpha \#* P$  and  $bn \alpha \#* Q$  and  $bn \alpha \#* R$ 
  by simp+
  from  $\langle \Psi \triangleright Q \parallel R \mapsto \alpha \prec QR \rangle \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* R \rangle \langle bn \alpha \#* \rangle$ 
  subject  $\alpha$ 
  show ?case

```

proof(*induct rule: parCases*[**where** $C = (P, R)$])
case(*cPar1* $Q' A_R \Psi_R$)
from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **by** *simp*
have FrR : *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from $\langle A_R \#* \alpha \rangle \langle bn \alpha \#* R \rangle$ FrR **have** $bn \alpha \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)
from $FrR \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \rightsquigarrow[Rel] Q$
by(*blast intro: Sim PRelQ*)
moreover **have** $QTrans$: $\Psi \otimes \Psi_R \triangleright Q \mapsto \alpha \prec Q'$ **by** *fact*
ultimately obtain P' **where** $PTrans$: $\Psi \otimes \Psi_R \triangleright P \mapsto \alpha \prec P'$
and $P'RelQ'$: $(\Psi \otimes \Psi_R, P', Q') \in Rel$
using $\langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* \Psi_R \rangle \langle bn \alpha \#* P \rangle$
by(*force dest: simE*)
from $PTrans$ $QTrans$ $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* \alpha \rangle \langle bn \alpha \#* subject \alpha \rangle$
 $\langle distinct(bn \alpha) \rangle$ **have** $A_R \#* P'$ **and** $A_R \#* Q'$
by(*blast dest: freeFreshChainDerivative*)
from $PTrans$ $\langle bn \alpha \#* R \rangle$ FrR $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* \alpha \rangle$ **have** $\Psi \triangleright P$
 $\parallel R \mapsto \alpha \prec (P' \parallel R)$
by(*metis Par1*)
moreover **from** $P'RelQ'$ FrR $\langle A_R \#* \Psi \rangle \langle A_R \#* P' \rangle \langle A_R \#* Q' \rangle$ **have** $(\Psi, P'$
 $\parallel R, Q' \parallel R) \in Rel'$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cPar2* $R' A_Q \Psi_Q$)
from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by** *simp+*
obtain $A_P \Psi_P$ **where** FrP : *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* (\Psi, A_Q,$
 $\Psi_Q, \alpha, R)$
by(*rule freshFrame*)
then **have** $A_P \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* \Psi_Q$ **and** $A_P \#* \alpha$ **and** $A_P \#*$
 R
by *simp+*

have FrQ : *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_Q \#* P \rangle$ FrP $\langle A_P \#* A_Q \rangle$ **have** $A_Q \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)

from FrP FrQ $\langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle A_P \#* \alpha \rangle \langle A_Q \#* \alpha \rangle$
have $bn \alpha \#* \Psi_P$ **and** $bn \alpha \#* \Psi_Q$
by(*force dest: extractFrameFreshChain*)

obtain $A_R \Psi_R$ **where** FrR : *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **and** $A_R \#* (\Psi, P, Q,$
 $A_Q, A_P, \Psi_Q, \Psi_P, \alpha, R)$ **and** *distinct* A_R
by(*rule freshFrame*)
then **have** $A_R \#* \Psi$ **and** $A_R \#* P$ **and** $A_R \#* Q$ **and** $A_R \#* A_Q$ **and** $A_R \#*$
 A_P **and** $A_R \#* \Psi_Q$ **and** $A_R \#* \Psi_P$ **and** $A_R \#* \alpha$ **and** $A_R \#* R$
by *simp+*

from $\langle A_Q \#* R \rangle$ FrR $\langle A_R \#* A_Q \rangle$ **have** $A_Q \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)

from $\langle A_P \#* R \rangle \langle A_R \#* A_P \rangle \text{FrR}$ **have** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)

have $R\text{Trans}: \Psi \otimes \Psi_Q \triangleright R \mapsto \alpha \prec R'$ **by fact**
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
proof –

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle$
by(*metis frameIntAssociativity Commutativity FrameStatEqTrans frameInt-CompositionSym FrameStatEqSym*)

moreover from $\text{FrR} \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
have (*insertAssertion (extractFrame Q) (\Psi \otimes \Psi_R)*) \hookrightarrow_F (*insertAssertion (extractFrame P) (\Psi \otimes \Psi_R)*)
by(*blast intro: PRelQ StatImp*)

with $\text{FrP FrQ} \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle$ **using** *freshCompChain by auto*

moreover have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by(*metis frameIntAssociativity Commutativity FrameStatEqTrans frameInt-CompositionSym frameIntAssociativity[THEN FrameStatEqSym]*)

ultimately show *?thesis*
by(*rule FrameStatEqImpCompose*)

qed

ultimately have $\Psi \otimes \Psi_P \triangleright R \mapsto \alpha \prec R'$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle$
 $\langle A_P \#* \alpha \rangle \langle A_Q \#* \alpha \rangle$
 $\langle A_R \#* A_P \rangle \langle A_R \#* A_Q \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* \Psi \rangle \text{FrR} \langle \text{distinct } A_R \rangle$
by(*force intro: transferFrame*)

with $\langle \text{bn } \alpha \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* R \rangle \langle A_P \#* \alpha \rangle \text{FrP}$ **have** $\Psi \triangleright P \parallel R$
 $\mapsto \alpha \prec (P \parallel R')$
by(*force intro: Par2*)

moreover obtain $A_{R'} \Psi_{R'}$ **where** *extractFrame R' = $\langle A_{R'}, \Psi_{R'} \rangle$* **and** $A_{R'} \#* \Psi$
and $A_{R'} \#* P$ **and** $A_{R'} \#* Q$
by(*rule freshFrame[where C=(Ψ, P, Q)]*) *auto*

moreover from $R\text{Trans FrR} \langle \text{distinct } A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
 $\langle A_R \#* R \rangle \langle A_R \#* \alpha \rangle \langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* Q \rangle \langle \text{bn } \alpha \#* R \rangle \langle \text{bn } \alpha \#* \text{subject } \alpha \rangle \langle \text{distinct}(\text{bn } \alpha) \rangle$
obtain $p \Psi' A_{R'} \Psi_{R'}$ **where** $S: \text{set } p \subseteq \text{set}(\text{bn } \alpha) \times \text{set}(\text{bn}(p \cdot \alpha))$ **and** $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_{R'}$ **and** $\text{FrR}': \text{extractFrame } R' = \langle A_{R'}, \Psi_{R'} \rangle$
and $\text{bn}(p \cdot \alpha) \#* R$ **and** $\text{bn}(p \cdot \alpha) \#* \Psi$ **and** $\text{bn}(p \cdot \alpha) \#* P$ **and** $\text{bn}(p \cdot \alpha) \#* Q$ **and** $\text{bn}(p \cdot \alpha) \#* R$
and $A_{R'} \#* \Psi$ **and** $A_{R'} \#* P$ **and** $A_{R'} \#* Q$
apply –
apply (*rule expandFrame[where C=(Ψ, P, Q, R) and C'=(Ψ, P, Q, R)]*)
apply *simp+*

done

from $\langle A_R \#* \Psi \rangle$ **have** $(p \cdot A_R) \#* (p \cdot \Psi)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle bn \ \alpha \ #* \ \Psi \rangle \langle bn(p \cdot \alpha) \ #* \ \Psi \rangle S$ **have** $(p \cdot A_R) \#* \Psi$ **by** *simp*
from $\langle A_R \#* P \rangle$ **have** $(p \cdot A_R) \#* (p \cdot P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle bn \ \alpha \ #* \ P \rangle \langle bn(p \cdot \alpha) \ #* \ P \rangle S$ **have** $(p \cdot A_R) \#* P$ **by** *simp*
from $\langle A_R \#* Q \rangle$ **have** $(p \cdot A_R) \#* (p \cdot Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle bn \ \alpha \ #* \ Q \rangle \langle bn(p \cdot \alpha) \ #* \ Q \rangle S$ **have** $(p \cdot A_R) \#* Q$ **by** *simp*

from *FrR* **have** $(p \cdot extractFrame \ R) = p \cdot \langle A_R, \Psi_R \rangle$ **by** *simp*
with $\langle bn \ \alpha \ #* \ R \rangle \langle bn(p \cdot \alpha) \ #* \ R \rangle S$ **have** $extractFrame \ R = \langle (p \cdot A_R), (p \cdot \Psi_R) \rangle$
by(*simp add: eqts*)

with $\langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle$ **have** $(\Psi \otimes (p \cdot \Psi_R), P, Q) \in Rel$
by(*metis PRelQ*)

then **have** $(\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', P, Q) \in Rel$ **by**(*rule Ext*)
with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', P, Q) \in Rel$ **by**(*blast intro: C3 Associativity compositionSym*)
with *FrR'* $\langle A_R' \#* \Psi \rangle \langle A_R' \#* P \rangle \langle A_R' \#* Q \rangle$ **have** $(\Psi, P \parallel R', Q \parallel R') \in Rel'$
by(*metis C1*)
ultimately **show** *?case* **by** *blast*

next
case(*cComm1* $\Psi_R \ M \ N \ Q' \ A_Q \ \Psi_Q \ K \ xvec \ R' \ A_R$)
have *FrQ*: $extractFrame \ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by** *simp+*

have *FrR*: $extractFrame \ R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* R$ **by** *simp+*

from $\langle xvec \#* (P, R) \rangle$ **have** $xvec \#* P$ **and** $xvec \#* R$ **by** *simp+*

obtain $A_P \ \Psi_P$ **where** *FrP*: $extractFrame \ P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* (\Psi, A_Q, \Psi_Q, A_R, M, N, K, R, P, xvec)$ **and** *distinct* A_P
by(*rule freshFrame*)
then **have** $A_P \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* \Psi_Q$ **and** $A_P \#* M$ **and** $A_P \#* R$
and $A_P \#* N$ **and** $A_P \#* K$ **and** $A_P \#* A_R$ **and** $A_P \#* P$ **and** $A_P \#* xvec$
by *simp+*

have *QTrans*: $\Psi \otimes \Psi_R \triangleright Q \mapsto M(|N|) \triangleleft Q'$ **and** *RTrans*: $\Psi \otimes \Psi_Q \triangleright R \mapsto K(|\nu * xvec|) \triangleleft R'$
and *MeqK*: $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$ **by** *fact+*

from *FrP* *FrR* $\langle A_Q \#* P \rangle \langle A_P \#* R \rangle \langle A_R \#* P \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* A_R \rangle \langle A_P \#* xvec \rangle \langle xvec \#* P \rangle$

have $A_P \#* \Psi_R$ **and** $A_Q \#* \Psi_P$ **and** $A_R \#* \Psi_P$ **and** $vec \#* \Psi_P$
by(*auto dest!*: *extractFrameFreshChain*)**+**

from $RTrans FrR \langle distinct A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* vec \rangle \langle vec \#* R \rangle \langle vec \#* Q \rangle \langle vec \#* \Psi \rangle \langle vec \#* \Psi_Q \rangle \langle A_R \#* Q \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_Q \rangle \langle vec \#* K \rangle \langle A_R \#* K \rangle \langle A_R \#* N \rangle \langle A_R \#* R \rangle \langle vec \#* R \rangle \langle A_R \#* P \rangle \langle vec \#* P \rangle \langle A_P \#* A_R \rangle \langle A_P \#* vec \rangle$
 $\langle A_Q \#* A_R \rangle \langle A_Q \#* vec \rangle \langle A_R \#* \Psi_P \rangle \langle vec \#* \Psi_P \rangle \langle distinct vec \rangle \langle vec \#* M \rangle$

obtain $p \Psi' A_R' \Psi_R'$ **where** $S: set p \subseteq set vec \times set(p \cdot vec)$ **and** $FrR': extractFrame R' = \langle A_R', \Psi_R' \rangle$
and $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \#* Q$ **and** $A_R' \#* \Psi$ **and** $(p \cdot vec) \#* \Psi$
and $(p \cdot vec) \#* Q$ **and** $(p \cdot vec) \#* \Psi_Q$ **and** $(p \cdot vec) \#* K$ **and** $(p \cdot vec) \#* R$
and $(p \cdot vec) \#* P$ **and** $(p \cdot vec) \#* A_P$ **and** $(p \cdot vec) \#* A_Q$ **and** $(p \cdot vec) \#* \Psi_P$
and $A_R' \#* P$ **and** $A_R' \#* N$
by (*subst expandFrame*[**where** $C=(\Psi, Q, \Psi_Q, K, R, P, A_P, A_Q, \Psi_P)$ **and** $C'=(\Psi, Q, \Psi_Q, K, R, P, A_P, A_Q, \Psi_P)$]) *simp***+**

from $\langle A_R \#* \Psi \rangle$ **have** $(p \cdot A_R) \#* (p \cdot \Psi)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle vec \#* \Psi \rangle \langle (p \cdot vec) \#* \Psi \rangle S$ **have** $(p \cdot A_R) \#* \Psi$ **by** *simp*
from $\langle A_R \#* P \rangle$ **have** $(p \cdot A_R) \#* (p \cdot P)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle vec \#* P \rangle \langle (p \cdot vec) \#* P \rangle S$ **have** $(p \cdot A_R) \#* P$ **by** *simp*
from $\langle A_R \#* Q \rangle$ **have** $(p \cdot A_R) \#* (p \cdot Q)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle vec \#* Q \rangle \langle (p \cdot vec) \#* Q \rangle S$ **have** $(p \cdot A_R) \#* Q$ **by** *simp*
from $\langle A_R \#* R \rangle$ **have** $(p \cdot A_R) \#* (p \cdot R)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle vec \#* R \rangle \langle (p \cdot vec) \#* R \rangle S$ **have** $(p \cdot A_R) \#* R$ **by** *simp*
from $\langle A_R \#* K \rangle$ **have** $(p \cdot A_R) \#* (p \cdot K)$ **by**(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle vec \#* K \rangle \langle (p \cdot vec) \#* K \rangle S$ **have** $(p \cdot A_R) \#* K$ **by** *simp*

from $\langle A_P \#* vec \rangle \langle (p \cdot vec) \#* A_P \rangle \langle A_P \#* M \rangle S$ **have** $A_P \#* (p \cdot M)$
by(*simp add: freshChainSimps*)
from $\langle A_Q \#* vec \rangle \langle (p \cdot vec) \#* A_Q \rangle \langle A_Q \#* M \rangle S$ **have** $A_Q \#* (p \cdot M)$
by(*simp add: freshChainSimps*)
from $\langle A_P \#* vec \rangle \langle (p \cdot vec) \#* A_P \rangle \langle A_P \#* A_R \rangle S$ **have** $(p \cdot A_R) \#* A_P$
by(*simp add: freshChainSimps*)
from $\langle A_Q \#* vec \rangle \langle (p \cdot vec) \#* A_Q \rangle \langle A_Q \#* A_R \rangle S$ **have** $(p \cdot A_R) \#* A_Q$
by(*simp add: freshChainSimps*)

from $QTrans S \langle vec \#* Q \rangle \langle (p \cdot vec) \#* Q \rangle$ **have** $(p \cdot (\Psi \otimes \Psi_R)) \triangleright Q \mapsto (p \cdot M)(N) \prec Q'$
using *inputPermFrameSubject name-list-set-fresh* **by** *blast*
with $\langle vec \#* \Psi \rangle \langle (p \cdot vec) \#* \Psi \rangle S$ **have** $QTrans: (\Psi \otimes (p \cdot \Psi_R)) \triangleright Q \mapsto$

$(p \cdot M)(\downarrow N) \prec Q'$
by(*simp add: eqvts*)

from *FrR* **have** $(p \cdot \text{extractFrame } R) = p \cdot \langle A_R, \Psi_R \rangle$ **by** *simp*
with $\langle \text{xvec} \#* R \rangle \langle (p \cdot \text{xvec}) \#* R \rangle S$ **have** *FrR*: $\text{extractFrame } R = \langle (p \cdot A_R), (p \cdot \Psi_R) \rangle$
by(*simp add: eqvts*)

note *RTrans FrR*
moreover from *FrR* $\langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle$ **have** $\Psi \otimes (p \cdot \Psi_R) \triangleright P \rightsquigarrow[\text{Rel}] Q$
by(*metis Sim PRelQ*)
with *QTrans* **obtain** P' **where** *PTrans*: $\Psi \otimes (p \cdot \Psi_R) \triangleright P \mapsto (p \cdot M)(\downarrow N) \prec P'$ **and** *P'RelQ'*: $(\Psi \otimes (p \cdot \Psi_R), P', Q') \in \text{Rel}$
by(*force dest: simE*)
from *PTrans* *QTrans* $\langle A_R' \#* P \rangle \langle A_R' \#* Q \rangle \langle A_R' \#* N \rangle$ **have** $A_R' \#* P'$ **and** $A_R' \#* Q'$
by(*blast dest: inputFreshChainDerivative*)+

note *PTrans*
moreover from *MeqK* **have** $(p \cdot (\Psi \otimes \Psi_Q \otimes \Psi_R)) \vdash (p \cdot M) \leftrightarrow (p \cdot K)$
by(*rule chanEqClosed*)
with $\langle \text{xvec} \#* \Psi \rangle \langle (p \cdot \text{xvec}) \#* \Psi \rangle \langle \text{xvec} \#* \Psi_Q \rangle \langle (p \cdot \text{xvec}) \#* \Psi_Q \rangle \langle \text{xvec} \#* K \rangle \langle (p \cdot \text{xvec}) \#* K \rangle S$
have *MeqK*: $\Psi \otimes \Psi_Q \otimes (p \cdot \Psi_R) \vdash (p \cdot M) \leftrightarrow K$ **by**(*simp add: eqvts*)

moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle$
proof –
have $\langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* $\langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle$
have (*insertAssertion* (*extractFrame* Q) $(\Psi \otimes (p \cdot \Psi_R))$) \hookrightarrow_F (*insertAssertion* (*extractFrame* P) $(\Psi \otimes (p \cdot \Psi_R))$)
by(*metis PRelQ StatImp*)
with *FrP* *FrQ* $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle \langle A_P \#* \text{xvec} \rangle \langle (p \cdot \text{xvec}) \#* A_P \rangle \langle A_Q \#* \text{xvec} \rangle \langle (p \cdot \text{xvec}) \#* A_Q \rangle S$
have $\langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P \rangle$ **using** *freshCompChain*
by(*simp add: freshChainSimps*)
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R) \rangle \simeq_F \langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)

qed
moreover note *FrP* *FrQ* $\langle \text{distinct } A_P \rangle$
moreover from $\langle \text{distinct } A_R \rangle$ **have** $\text{distinct}(p \cdot A_R)$ **by** *simp*

moreover note $\langle (p \cdot A_R) \#* A_P \rangle \langle (p \cdot A_R) \#* A_Q \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle \langle (p \cdot A_R) \#* R \rangle \langle (p \cdot A_R) \#* K \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* R \rangle \langle A_P \#* P \rangle \langle A_P \#* (p \cdot M) \rangle \langle A_Q \#* R \rangle \langle A_Q \#* (p \cdot M) \rangle$
 $\langle A_P \#* xvec \rangle \langle xvec \#* P \rangle \langle A_P \#* R \rangle$
ultimately obtain K' **where** $\Psi \otimes \Psi_P \triangleright R \mapsto K'(\nu*xvec)\langle N \rangle \prec R'$ **and** Ψ
 $\otimes \Psi_P \otimes (p \cdot \Psi_R) \vdash (p \cdot M) \leftrightarrow K'$ **and** $(p \cdot A_R) \#* K'$
using *comm1Aux* **by** *blast*

with *PTrans FrP* **have** $\Psi \triangleright P \parallel R \mapsto \tau \prec (\nu*xvec)\langle P' \parallel R' \rangle$ **using** *FrR* $\langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* R \rangle$
 $\langle xvec \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* R \rangle \langle A_P \#* (p \cdot M) \rangle \langle (p \cdot A_R) \#* K' \rangle \langle (p \cdot A_R) \#* A_P \rangle$
by (*intro Comm1*) (*assumption* | *simp*)+

moreover from $P'RelQ'$ **have** $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)

with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)

with *FrR'* $\langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* \Psi \rangle$ **have** $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$

by(*metis C1*)

with $\langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*xvec)\langle P' \parallel R' \rangle, (\nu*xvec)\langle Q' \parallel R' \rangle) \in Rel'$

by(*metis C2*)

ultimately show *?case* **by** *blast*

next

case(*cComm2* $\Psi_R M xvec N Q' A_Q \Psi_Q K R' A_R$)

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*

from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by** *simp*+

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*

from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* R$ **by** *simp*+

from $\langle xvec \#* (P, R) \rangle$ **have** $xvec \#* P$ **and** $xvec \#* R$ **by** *simp*+

obtain $A_P \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* (\Psi, A_Q, \Psi_Q, A_R, M, N, K, R, P, xvec)$ **and** *distinct* A_P

by(*rule freshFrame*)

then have $A_P \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* \Psi_Q$ **and** $A_P \#* M$ **and** $A_P \#* R$

and $A_P \#* N$ $A_P \#* K$ **and** $A_P \#* A_R$ **and** $A_P \#* P$ **and** $A_P \#* xvec$

by *simp*+

from *FrP* *FrR* $\langle A_Q \#* P \rangle \langle A_P \#* R \rangle \langle A_R \#* P \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* A_R \rangle \langle A_P \#* xvec \rangle \langle xvec \#* P \rangle$

have $A_P \#* \Psi_R$ **and** $A_Q \#* \Psi_P$ **and** $A_R \#* \Psi_P$ **and** $xvec \#* \Psi_P$

by(*auto dest!*: *extractFrameFreshChain*)+

have *QTrans*: $\Psi \otimes \Psi_R \triangleright Q \mapsto M(\nu*xvec)\langle N \rangle \prec Q'$ **by** *fact*

note $\langle \Psi \otimes \Psi_Q \triangleright R \mapsto K(\lfloor N \rfloor) \prec R' \rangle$ *FrR* $\langle \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$
moreover from *FrR* $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P$
 \rightsquigarrow [*Rel*] *Q* **by** (*metis PRelQ Sim*)
with *QTrans* **obtain** *P'* **where** *PTrans*: $\Psi \otimes \Psi_R \triangleright P \mapsto M(\nu * \text{vec})(\lfloor N \rfloor) \prec$
P' **and** *P'RelQ'*: $(\Psi \otimes \Psi_R, P', Q') \in \text{Rel}$
using $\langle \text{vec} \#* \Psi \rangle \langle \text{vec} \#* \Psi_R \rangle \langle \text{vec} \#* P \rangle$
by (*force dest: simE*)
from *PTrans* *QTrans* $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* \text{vec} \rangle \langle \text{vec} \#* M \rangle \langle \text{distinct}$
 $\text{vec} \rangle$ **have** $A_R \#* P'$ **and** $A_R \#* Q'$
by (*blast dest: outputFreshChainDerivative*) +
note *PTrans* $\langle \Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
proof –
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by (*metis frameResChainPres frameNilStatEq Commutativity AssertionStatE-*
qTrans Composition Associativity)
moreover from *FrR* $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
have (*insertAssertion (extractFrame Q) (Psi ot Psi_R)*) \hookrightarrow_F (*insertAssertion*
(*extractFrame P*) $(\Psi \otimes \Psi_R)$)
by (*metis PRelQ StatImp*)
with *FrP FrQ* $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle$ **using** *freshCom-*
pChain **by** *simp*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle$
by (*metis frameResChainPres frameNilStatEq Commutativity AssertionStatE-*
qTrans Composition Associativity)
ultimately show *?thesis*
by (*metis FrameStatEqImpCompose*)
qed
moreover note *FrP FrQ* $\langle \text{distinct } A_P \rangle \langle \text{distinct } A_R \rangle$
moreover from $\langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$ **have** $A_R \#* A_P$ **and** $A_R \#* A_Q$ **by**
simp +
moreover note $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* K \rangle \langle A_P$
 $\#* \Psi \rangle \langle A_P \#* P \rangle$
 $\langle A_P \#* R \rangle \langle A_P \#* M \rangle \langle A_Q \#* R \rangle \langle A_Q \#* M \rangle \langle A_R \#* \text{vec} \rangle \langle \text{vec} \#* M \rangle$
ultimately obtain *K'* **where** $\Psi \otimes \Psi_P \triangleright R \mapsto K'(\lfloor N \rfloor) \prec R'$ **and** $\Psi \otimes \Psi_P \otimes$
 $\Psi_R \vdash M \leftrightarrow K'$ **and** $A_R \#* K'$
using *comm2Aux* **by** *blast*

with *PTrans* *FrP* **have** $\Psi \triangleright P \parallel R \mapsto \tau \prec (\nu * \text{vec})(P' \parallel R')$ **using** *FrR* $\langle A_R$
 $\#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$ **and** $\langle \text{vec} \#* R \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P$
 $\#* R \rangle \langle A_P \#* A_R \rangle \langle A_P \#* M \rangle \langle A_R \#* K' \rangle$
by (*force intro: Comm2*)

moreover from $\langle \Psi \otimes \Psi_P \triangleright R \mapsto K'(\lfloor N \rfloor) \prec R' \rangle$ *FrR* $\langle \text{distinct } A_R \rangle \langle A_R \#* \Psi \rangle$
 $\langle A_R \#* R \rangle \langle A_R \#* P' \rangle \langle A_R \#* Q' \rangle \langle A_R \#* N \rangle \langle A_R \#* K' \rangle$
obtain $\Psi' A_R' \Psi_R'$ **where** *ReqR'*: $\Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** *FrR'*: *extractFrame*
 $R' = \langle A_R', \Psi_R' \rangle$

and $A_R' \#* \Psi$ **and** $A_R' \#* P'$ **and** $A_R' \#* Q'$
by(*auto intro: expandFrame[where $C=(\Psi, P', Q')$ and $C'=\Psi$]*)

from $P'RelQ'$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with $ReqR'$ **have** $(\Psi \otimes \Psi_{R'}, P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with $FrR' \langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* \Psi \rangle$ **have** $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)
with $\langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*xvec)(P' \parallel R'), (\nu*xvec)(Q' \parallel R')) \in Rel'$
by(*metis C2*)
ultimately show *?case* **by** *blast*

next
case(*cBrMerge $\Psi_R M N Q' A_Q \Psi_Q R' A_R$*)
have $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by** *simp+*

have $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* R$ **by** *simp+*

obtain $A_P \Psi_P$ **where** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* (\Psi, A_Q, \Psi_Q, A_R, M, N, R, P)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* \Psi_Q$ **and** $A_P \#* M$ **and** $A_P \#* R$
and $A_P \#* N$ **and** $A_P \#* A_R$ **and** $A_P \#* P$
by *simp+*

from $FrP FrR \langle A_Q \#* P \rangle \langle A_P \#* R \rangle \langle A_R \#* P \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* A_R \rangle$
have $A_P \#* \Psi_R$ **and** $A_Q \#* \Psi_P$ **and** $A_R \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)+

have $QTrans: \Psi \otimes \Psi_R \triangleright Q \mapsto_{iM} \langle N \rangle \prec Q'$ **by** *fact*

have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto_{iM} \langle N \rangle \prec R'$ **by** *fact*

from $FrR \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \rightsquigarrow[Rel] Q$
by(*metis PRelQ Sim*)
with $QTrans$ **obtain** P' **where** $PTrans: \Psi \otimes \Psi_R \triangleright P \mapsto_{iM} \langle N \rangle \prec P'$ **and**
 $P'RelQ': (\Psi \otimes \Psi_R, P', Q') \in Rel$
by(*force dest: simE*)
from $PTrans QTrans \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* N \rangle$ **have** $A_R \#* P'$ **and**
 $A_R \#* Q'$
by(*blast dest: brinputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
proof –
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatE-*)

qTrans Composition Associativity)
moreover from $FrR \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
have $(insertAssertion (extractFrame Q) (\Psi \otimes \Psi_R)) \hookrightarrow_F (insertAssertion (extractFrame P) (\Psi \otimes \Psi_R))$
by(*metis PRelQ StatImp*)
with $FrP FrQ \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle$ **using** *freshCompChain by simp*
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
with $RTrans \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle distinct A_R \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_P \#* R \rangle \langle A_P \#* M \rangle$
 $\langle A_Q \#* R \rangle \langle A_Q \#* M \rangle$
have $\Psi \otimes \Psi_P \triangleright R \mapsto_{iM} \langle N \rangle \prec R'$
using *brCommInAux freshChainSym by blast*

with $PTrans FrP$ **have** *Transition: $\Psi \triangleright P \parallel R \mapsto_{iM} \langle N \rangle \prec (P' \parallel R')$* **using**
 $FrR \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* R \rangle \langle A_P \#* A_R \rangle$
 $\langle A_P \#* M \rangle \langle A_R \#* M \rangle$
by(*force intro: BrMerge*)

from $\langle \Psi \otimes \Psi_P \triangleright R \mapsto_{iM} \langle N \rangle \prec R' \rangle FrR \langle distinct A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* P' \rangle \langle A_R \#* Q' \rangle \langle A_R \#* N \rangle \langle A_R \#* M \rangle$
obtain $\Psi' AR' \Psi_R'$ **where** $ReqR': \Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** $FrR': extractFrame R' = \langle A_R', \Psi_R' \rangle$
and $A_R' \#* \Psi$ **and** $A_R' \#* P'$ **and** $A_R' \#* Q'$
by(*auto intro: expandFrame[where C=(Ψ, P', Q') and C'= Ψ]*)

from $P'RelQ'$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with $ReqR'$ **have** $(\Psi \otimes \Psi_R', P', Q') \in Rel$ **by**(*metis C3 Associativity compositionSym*)
with $FrR' \langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* \Psi \rangle$ **have** *Relation: $(\Psi, P' \parallel R', Q' \parallel R') \in Rel'$*
by(*metis C1*)
show *?case using Transition Relation*
by *blast*
next
case(*cBrComm1 $\Psi_R M N Q' A_Q \Psi_Q xvec R' A_R$*)
have $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by** *simp+*

have $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact*
from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* R$ **by** *simp+*

from $\langle \downarrow M(\nu * xvec) \rangle \langle N \rangle = \alpha$ **have** $xvec = bn \ \alpha$
by(*auto simp add: action.inject*)
from $\langle xvec = bn \ \alpha \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ R \rangle$
have $xvec \ \#* \ P$ **and** $xvec \ \#* \ R$ **by** *simp+*

obtain $A_P \ \Psi_P$ **where** $FrP: extractFrame \ P = \langle A_P, \Psi_P \rangle$ **and** $A_P \ \#* \ (\Psi, A_Q,$
 $\Psi_Q, A_R, M, N, R, P, xvec)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \ \#* \ \Psi$ **and** $A_P \ \#* \ A_Q$ **and** $A_P \ \#* \ \Psi_Q$ **and** $A_P \ \#* \ M$ **and** $A_P \ \#* \ R$
and $A_P \ \#* \ N$ **and** $A_P \ \#* \ A_R$ **and** $A_P \ \#* \ P$ **and** $A_P \ \#* \ xvec$
by *simp+*

from $FrP \ FrR \ \langle A_Q \ \#* \ P \rangle \langle A_P \ \#* \ R \rangle \langle A_R \ \#* \ P \rangle \langle A_P \ \#* \ A_Q \rangle \langle A_P \ \#* \ A_R \rangle$
have $A_P \ \#* \ \Psi_R$ **and** $A_Q \ \#* \ \Psi_P$ **and** $A_R \ \#* \ \Psi_P$
by(*auto dest: extractFrameFreshChain*)+

from $\langle A_P \ \#* \ xvec \rangle \langle xvec \ \#* \ P \rangle \ FrP$ **have** $xvec \ \#* \ \Psi_P$
by(*auto dest: extractFrameFreshChain*)

have $QTrans: \Psi \otimes \Psi_R \triangleright Q \mapsto \downarrow M(N) \prec Q'$ **by** *fact*

have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto \downarrow M(\nu * xvec) \langle N \rangle \prec R'$ **by** *fact*

from $RTrans \ FrR \ \langle distinct \ A_R \rangle \langle A_R \ \#* \ R \rangle \langle A_R \ \#* \ xvec \rangle \langle xvec \ \#* \ R \rangle \langle xvec \ \#* \ Q \rangle$
 $\langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ \Psi_Q \rangle \langle A_R \ \#* \ Q \rangle$
 $\langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ \Psi_Q \rangle \langle A_R \ \#* \ M \rangle \langle A_R \ \#* \ N \rangle \langle A_R \ \#* \ R \rangle \langle xvec \ \#* \ R \rangle \langle A_R \ \#* \ P \rangle$
 $\langle xvec \ \#* \ P \rangle \langle A_P \ \#* \ A_R \rangle \langle A_P \ \#* \ xvec \rangle$
 $\langle A_Q \ \#* \ A_R \rangle \langle A_Q \ \#* \ xvec \rangle \langle A_R \ \#* \ \Psi_P \rangle \langle xvec \ \#* \ \Psi_P \rangle \langle distinct \ xvec \rangle \langle xvec \ \#* \ M \rangle$

obtain $p \ \Psi' \ A_R' \ \Psi_R'$ **where** $S: set \ p \subseteq set \ xvec \times set \ (p \cdot xvec)$ **and** $FrR':$
 $extractFrame \ R' = \langle A_R', \Psi_R' \rangle$
and $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \ \#* \ Q$ **and** $A_R' \ \#* \ \Psi$ **and** $(p \cdot xvec) \ \#* \ \Psi$
and $(p \cdot xvec) \ \#* \ Q$ **and** $(p \cdot xvec) \ \#* \ \Psi_Q$ **and** $(p \cdot xvec) \ \#* \ R$ **and** $(p \cdot xvec)$
 $\#* \ M$
and $(p \cdot xvec) \ \#* \ P$ **and** $(p \cdot xvec) \ \#* \ A_P$ **and** $(p \cdot xvec) \ \#* \ A_Q$ **and** $(p \cdot xvec)$
 $\#* \ \Psi_P$
and $A_R' \ \#* \ P$ **and** $A_R' \ \#* \ N$
by(*auto intro: expandFrame[where C=(\Psi, Q, \Psi_Q, R, P, M, A_P, A_Q, \Psi_P)*)
and $C'=(\Psi, Q, \Psi_Q, R, P, M, A_P, A_Q, \Psi_P)$)

from $\langle A_R \ \#* \ \Psi \rangle$ **have** $(p \cdot A_R) \ \#* \ (p \cdot \Psi)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \ \#* \ \Psi \rangle \langle (p \cdot xvec) \ \#* \ \Psi \rangle \ S$ **have** $(p \cdot A_R) \ \#* \ \Psi$ **by** *simp*
from $\langle A_R \ \#* \ \Psi_P \rangle$ **have** $(p \cdot A_R) \ \#* \ (p \cdot \Psi_P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \ \#* \ \Psi_P \rangle \langle (p \cdot xvec) \ \#* \ \Psi_P \rangle \ S$ **have** $(p \cdot A_R) \ \#* \ \Psi_P$ **by** *simp*
from $\langle A_R \ \#* \ \Psi_Q \rangle$ **have** $(p \cdot A_R) \ \#* \ (p \cdot \Psi_Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle xvec \#* \Psi_Q \rangle \langle (p \cdot xvec) \#* \Psi_Q \rangle S$ **have** $(p \cdot A_R) \#* \Psi_Q$ **by** *simp*
from $\langle A_R \#* M \rangle$ **have** $(p \cdot A_R) \#* (p \cdot M)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \#* M \rangle \langle (p \cdot xvec) \#* M \rangle S$ **have** $(p \cdot A_R) \#* M$ **by** *simp*
from $\langle A_R \#* P \rangle$ **have** $(p \cdot A_R) \#* (p \cdot P)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle S$ **have** $(p \cdot A_R) \#* P$ **by** *simp*
from $\langle A_R \#* Q \rangle$ **have** $(p \cdot A_R) \#* (p \cdot Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle S$ **have** $(p \cdot A_R) \#* Q$ **by** *simp*
from $\langle A_R \#* R \rangle$ **have** $(p \cdot A_R) \#* (p \cdot R)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle xvec \#* R \rangle \langle (p \cdot xvec) \#* R \rangle S$ **have** $(p \cdot A_R) \#* R$ **by** *simp*

from $\langle A_P \#* xvec \rangle \langle (p \cdot xvec) \#* A_P \rangle \langle A_P \#* M \rangle S$ **have** $A_P \#* (p \cdot M)$
by(*simp add: freshChainSimps*)
from $\langle A_Q \#* xvec \rangle \langle (p \cdot xvec) \#* A_Q \rangle \langle A_Q \#* M \rangle S$ **have** $A_Q \#* (p \cdot M)$
by(*simp add: freshChainSimps*)
from $\langle A_P \#* xvec \rangle \langle (p \cdot xvec) \#* A_P \rangle \langle A_P \#* A_R \rangle S$ **have** $(p \cdot A_R) \#* A_P$
by(*simp add: freshChainSimps*)
from $\langle A_Q \#* xvec \rangle \langle (p \cdot xvec) \#* A_Q \rangle \langle A_Q \#* A_R \rangle S$ **have** $(p \cdot A_R) \#* A_Q$
by(*simp add: freshChainSimps*)

from $QTrans S \langle xvec \#* Q \rangle \langle (p \cdot xvec) \#* Q \rangle$ **have** $(p \cdot (\Psi \otimes \Psi_R)) \triangleright Q \mapsto$
 $i(p \cdot M)(\downarrow N) \prec Q'$
using *brinputPermFrameSubject name-list-set-fresh by blast*
with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle S$ **have** $QTrans: (\Psi \otimes (p \cdot \Psi_R)) \triangleright Q \mapsto$
 $i(p \cdot M)(\downarrow N) \prec Q'$
by(*simp add: eqts*)

from FrR **have** $(p \cdot extractFrame R) = p \cdot \langle A_R, \Psi_R \rangle$ **by** *simp*
with $\langle xvec \#* R \rangle \langle (p \cdot xvec) \#* R \rangle S$ **have** $FrR: extractFrame R = \langle (p \cdot A_R),$
 $(p \cdot \Psi_R) \rangle$
by(*simp add: eqts*)

from $FrR \langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle$ **have** $\Psi \otimes (p \cdot \Psi_R)$
 $\triangleright P \rightsquigarrow[Rel] Q$
by(*metis Sim PRelQ*)
with $QTrans$ **obtain** P' **where** $PTrans: \Psi \otimes (p \cdot \Psi_R) \triangleright P \mapsto i(p \cdot M)(\downarrow N)$
 $\prec P'$ **and** $P'RelQ': (\Psi \otimes (p \cdot \Psi_R), P', Q') \in Rel$
by(*force dest: simE*)
with $QTrans \langle A_{R'} \#* P \rangle \langle A_{R'} \#* Q \rangle \langle A_{R'} \#* N \rangle$ **have** $A_{R'} \#* P'$ **and** $A_{R'} \#*$
 Q'
by(*blast dest: brinputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R) \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle$
proof –
have $\langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes (p \cdot \Psi_R) \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatE*)

qTrans Composition Associativity
moreover from $FrR \langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle$
have $(insertAssertion (extractFrame Q) (\Psi \otimes (p \cdot \Psi_R))) \hookrightarrow_F (insertAssertion (extractFrame P) (\Psi \otimes (p \cdot \Psi_R)))$
by(*metis PRelQ StatImp*)
with $FrP FrQ \langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle \langle A_P \#* xvec \rangle$
 $\langle (p \cdot xvec) \#* A_P \rangle \langle A_Q \#* xvec \rangle \langle (p \cdot xvec) \#* A_Q \rangle S$
have $\langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_P \rangle$ **using**
freshCompChain
by(*simp add: freshChainSimps*)
moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes (p \cdot \Psi_R) \rangle \simeq_F \langle A_Q, (\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi_Q \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEq*)
qTrans Composition Associativity
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
moreover note $RTrans FrR FrP FrQ \langle distinct A_P \rangle$
moreover from $\langle distinct A_R \rangle$ **have** $distinct(p \cdot A_R)$ **by** *simp*
moreover note $\langle (p \cdot A_R) \#* A_P \rangle \langle (p \cdot A_R) \#* A_Q \rangle \langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi_P \rangle \langle (p \cdot A_R) \#* \Psi_Q \rangle \langle (p \cdot A_R) \#* M \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* Q \rangle \langle (p \cdot A_R) \#* R \rangle$
 $\langle A_P \#* \Psi \rangle \langle A_P \#* R \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* (p \cdot M) \rangle \langle A_Q \#* R \rangle \langle A_P \#* xvec \rangle \langle xvec \#* P \rangle \langle A_P \#* R \rangle$
ultimately have $\Psi \otimes \Psi_P \triangleright R \mapsto_i M(\nu * xvec)\langle N \rangle \prec R'$
using *brCommOutAux* **by** *blast*
with $S \langle xvec \#* M \rangle \langle (p \cdot xvec) \#* M \rangle$ **have** $\Psi \otimes \Psi_P \triangleright R \mapsto_i (p \cdot M)(\nu * xvec)\langle N \rangle \prec R'$ **by** *simp*

with $PTrans FrP S \langle xvec \#* M \rangle \langle (p \cdot xvec) \#* M \rangle \langle (p \cdot A_R) \#* M \rangle$ **have** $\Psi \triangleright P \parallel R \mapsto_i (p \cdot M)(\nu * xvec)\langle N \rangle \prec (P' \parallel R')$ **using** $FrR \langle (p \cdot A_R) \#* \Psi \rangle \langle (p \cdot A_R) \#* P \rangle \langle (p \cdot A_R) \#* R \rangle$
 $\langle xvec \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* R \rangle \langle A_P \#* (p \cdot M) \rangle \langle (p \cdot A_R) \#* A_P \rangle$
by(*intro BrComm1*) (*assumption* | *simp*)+

with $S \langle xvec \#* M \rangle \langle (p \cdot xvec) \#* M \rangle$
have $Transition: \Psi \triangleright P \parallel R \mapsto_i M(\nu * xvec)\langle N \rangle \prec (P' \parallel R')$ **by** *simp*

from $P'RelQ'$ **have** $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', P', Q') \in Rel$ **by**(*rule Ext*)
with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, P', Q') \in Rel$ **by**(*metis C3 Associativity composition.Sym*)
with $FrR' \langle A_{R'} \#* P' \rangle \langle A_{R'} \#* Q' \rangle \langle A_{R'} \#* \Psi \rangle$ **have** $Relation: (\Psi, P' \parallel R', Q' \parallel R') \in Rel'$
by(*metis C1*)

show *?case* **using** *Transition Relation*
by *blast*
next

case(*cBrComm2* $\Psi_R M \text{vec } N Q' A_Q \Psi_Q R' A_R$)
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by fact**
from $\langle A_Q \#* (P, R) \rangle$ **have** $A_Q \#* P$ **and** $A_Q \#* R$ **by simp+**

have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by fact**
from $\langle A_R \#* (P, R) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* R$ **by simp+**
from $\langle \text{!}M(\nu * \text{vec}) \langle N \rangle = \alpha \rangle$ **have** $\text{vec} = \text{bn } \alpha$
by(*auto simp add: action.inject*)
from $\langle \text{vec} = \text{bn } \alpha \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* R \rangle$
have $\text{vec} \#* P$ **and** $\text{vec} \#* R$ **by simp+**

obtain $A_P \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* (\Psi, A_Q,$
 $\Psi_Q, A_R, M, N, R, P)$ **and** *distinct* A_P
by(*rule freshFrame*)
then have $A_P \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* \Psi_Q$ **and** $A_P \#* M$ **and** $A_P \#*$
 R
and $A_P \#* N$ **and** $A_P \#* A_R$ **and** $A_P \#* P$
by simp+

from *FrP FrR* $\langle A_Q \#* P \rangle \langle A_P \#* R \rangle \langle A_R \#* P \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* A_R \rangle$
have $A_P \#* \Psi_R$ **and** $A_Q \#* \Psi_P$ **and** $A_R \#* \Psi_P$
by(*auto dest: extractFrameFreshChain*)+

have *QTrans*: $\Psi \otimes \Psi_R \triangleright Q \mapsto \text{!}M(\nu * \text{vec}) \langle N \rangle \prec Q'$ **by fact**

have *RTrans*: $\Psi \otimes \Psi_Q \triangleright R \mapsto \text{!}M \langle N \rangle \prec R'$ **by fact**

from *FrR* $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$ **have** $\Psi \otimes \Psi_R \triangleright P \rightsquigarrow [\text{Rel}] Q$
by(*metis PRelQ Sim*)
from $\langle \Psi \otimes \Psi_R \triangleright P \rightsquigarrow [\text{Rel}] Q \rangle$ *QTrans* $\langle \text{vec} \#* \Psi \rangle \langle \text{vec} \#* \Psi_R \rangle \langle \text{vec} \#* P \rangle$
obtain P' **where** *PTrans*: $\Psi \otimes \Psi_R \triangleright P \mapsto \text{!}M(\nu * \text{vec}) \langle N \rangle \prec P'$ **and** $P' \text{Rel} Q'$:
 $(\Psi \otimes \Psi_R, P', Q') \in \text{Rel}$
by(*force dest: simE*)
from *PTrans QTrans* $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* N \rangle \langle A_R \#* \text{vec} \rangle \langle \text{vec} \#*$
 $M \rangle \langle \text{distinct } \text{vec} \rangle$ **have** $A_R \#* P'$ **and** $A_R \#* Q'$
by(*blast dest: broutputFreshChainDerivative*)+

have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
proof –
have $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
moreover from *FrR* $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
have (*insertAssertion (extractFrame Q) (Psi ot Psi_R)*) \hookrightarrow_F (*insertAssertion (extractFrame P) (Psi ot Psi_R)*)
by(*metis PRelQ StatImp*)
with *FrP FrQ* $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi_R \rangle$
have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle$ **using** *freshCom-
pChain by simp*

moreover have $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle$
by(*metis frameResChainPres frameNilStatEq Commutativity AssertionStatEqTrans Composition Associativity*)
ultimately show *?thesis*
by(*metis FrameStatEqImpCompose*)
qed
with *RTrans* $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \langle \text{distinct } A_R \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_P \#* R \rangle \langle A_P \#* M \rangle$
 $\langle A_Q \#* R \rangle \langle A_Q \#* M \rangle$
have $\Psi \otimes \Psi_P \triangleright R \mapsto_{\downarrow} M(\downarrow N) \prec R'$
using *brCommInAux freshChainSym* **by** *blast*

with *PTrans FrP* **have** *Transition*: $\Psi \triangleright P \parallel R \mapsto_{\downarrow} M(\downarrow \nu * \text{xvec}) \langle N \rangle \prec (P' \parallel R')$ **using** *FrR* $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* R \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* R \rangle \langle A_P \#* A_R \rangle$
 $\langle A_P \#* M \rangle \langle A_R \#* M \rangle \langle \text{xvec} \#* R \rangle$
by(*force intro: BrComm2*)

from $\langle \Psi \otimes \Psi_P \triangleright R \mapsto_{\downarrow} M(\downarrow N) \prec R' \rangle$ *FrR* $\langle \text{distinct } A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* P' \rangle \langle A_R \#* Q' \rangle \langle A_R \#* N \rangle \langle A_R \#* M \rangle$
obtain $\Psi' A_R' \Psi_R'$ **where** *ReqR'*: $\Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** *FrR'*: *extractFrame* $R' = \langle A_R', \Psi_R' \rangle$
and $A_R' \#* \Psi$ **and** $A_R' \#* P'$ **and** $A_R' \#* Q'$
by(*auto intro: expandFrame[where C=(Ψ, P', Q') and C'= Ψ]*)

from *P'RelQ'* **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', Q') \in \text{Rel}$ **by**(*rule Ext*)
with *ReqR'* **have** $(\Psi \otimes \Psi_{R'}, P', Q') \in \text{Rel}$ **by**(*metis C3 Associativity compositionSym*)
with *FrR'* $\langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* \Psi \rangle$ **have** *Relation*: $(\Psi, P' \parallel R', Q' \parallel R') \in \text{Rel}'$
by(*metis C1*)
show *?case* **using** *Transition Relation*
by *blast*
qed
qed

no-notation *relcomp* (*infixr* 0 75)

lemma *bangPres*:
fixes $\Psi \quad :: 'b$
and $P \quad :: ('a, 'b, 'c) \text{ psi}$
and $Q \quad :: ('a, 'b, 'c) \text{ psi}$
and $R \quad :: ('a, 'b, 'c) \text{ psi}$
and $\text{Rel} \quad :: ('b \times ('a, 'b, 'c) \text{ psi} \times ('a, 'b, 'c) \text{ psi}) \text{ set}$
and $\text{Rel}' \quad :: ('b \times ('a, 'b, 'c) \text{ psi} \times ('a, 'b, 'c) \text{ psi}) \text{ set}$

assumes $(\Psi, P, Q) \in \text{Rel}$
and *eqvt Rel'*

and *guarded P*
and *guarded Q*
and *cSim*: $\bigwedge \Psi' S T. (\Psi', S, T) \in Rel \implies \Psi' \triangleright S \rightsquigarrow[Rel] T$
and *cExt*: $\bigwedge \Psi' S T \Psi''. (\Psi', S, T) \in Rel \implies (\Psi' \otimes \Psi'', S, T) \in Rel$
and *cSym*: $\bigwedge \Psi' S T. (\Psi', S, T) \in Rel \implies (\Psi', T, S) \in Rel$
and *StatEq*: $\bigwedge \Psi' S T \Psi''. \llbracket (\Psi', S, T) \in Rel; \Psi' \simeq \Psi'' \rrbracket \implies (\Psi'', S, T) \in Rel$
and *Closed*: $\bigwedge \Psi' S T p. (\Psi', S, T) \in Rel \implies ((p::name\ prm) \cdot \Psi', p \cdot S, p \cdot T) \in Rel$
and *Assoc*: $\bigwedge \Psi' S T U. (\Psi', S \parallel (T \parallel U), (S \parallel T) \parallel U) \in Rel$
and *ParPres*: $\bigwedge \Psi' S T U. (\Psi', S, T) \in Rel \implies (\Psi', S \parallel U, T \parallel U) \in Rel$
and *FrameParPres*: $\bigwedge \Psi' \Psi_U S T U A_U. \llbracket (\Psi' \otimes \Psi_U, S, T) \in Rel; extractFrame\ U = \langle A_U, \Psi_U \rangle; A_U \#* \Psi'; A_U \#* S; A_U \#* T \rrbracket \implies$
 $(\Psi', U \parallel S, U \parallel T) \in Rel$
and *ResPres*: $\bigwedge \Psi' S T xvec. \llbracket (\Psi', S, T) \in Rel; xvec \#* \Psi' \rrbracket \implies (\Psi', (\nu * xvec) S, (\nu * xvec) T) \in Rel$
and *ScopeExt*: $\bigwedge xvec \Psi' S T. \llbracket xvec \#* \Psi'; xvec \#* T \rrbracket \implies (\Psi', (\nu * xvec)(S \parallel T), ((\nu * xvec) S) \parallel T) \in Rel$
and *Trans*: $\bigwedge \Psi' S T U. \llbracket (\Psi', S, T) \in Rel; (\Psi', T, U) \in Rel \rrbracket \implies (\Psi', S, U) \in Rel$
and *Compose*: $\bigwedge \Psi' S T U O. \llbracket (\Psi', S, T) \in Rel; (\Psi', T, U) \in Rel'; (\Psi', U, O) \in Rel' \rrbracket \implies (\Psi', S, O) \in Rel'$
and *C1*: $\bigwedge \Psi S T U. \llbracket (\Psi, S, T) \in Rel; guarded\ S; guarded\ T \rrbracket \implies (\Psi, U \parallel !S, U \parallel !T) \in Rel'$
and *Der*: $\bigwedge \Psi' S \alpha S' T. \llbracket \Psi' \triangleright !S \mapsto_{\alpha} \prec S'; (\Psi', S, T) \in Rel; bn\ \alpha \#* \Psi'; bn\ \alpha \#* S; bn\ \alpha \#* T; guarded\ T; bn\ \alpha \#* subject\ \alpha \rrbracket \implies$
 $\exists T' U O. \Psi' \triangleright !T \mapsto_{\alpha} \prec T' \wedge (\Psi', S', U \parallel !S) \in Rel \wedge (\Psi', T', O \parallel !T) \in Rel \wedge$
 $(\Psi', U, O) \in Rel \wedge ((supp\ U)::name\ set) \subseteq$
 $supp\ S' \wedge$
 $((supp\ O)::name\ set) \subseteq supp\ T'$

shows $\Psi \triangleright R \parallel !P \rightsquigarrow[Rel'] R \parallel !Q$
using $\langle eqvt\ Rel' \rangle$
proof(*induct rule: simI[of - - - ()]*)
case(*cSim* $\alpha\ RQ'$)
from $\langle bn\ \alpha \#* (R \parallel !P) \rangle \langle bn\ \alpha \#* (R \parallel !Q) \rangle$ **have** $bn\ \alpha \#* P$ **and** $bn\ \alpha \#* (!Q)$
and $bn\ \alpha \#* Q$ **and** $bn\ \alpha \#* R$ **by** *simp+*
from $\langle \Psi \triangleright R \parallel !Q \mapsto_{\alpha} \prec RQ' \rangle \langle bn\ \alpha \#* \Psi \rangle \langle bn\ \alpha \#* R \rangle \langle bn\ \alpha \#* !Q \rangle \langle bn\ \alpha \#* subject\ \alpha \rangle$ **show** *?case*
proof(*induct rule: parCases[where C=P]*)
case(*cPar1* $R' A_Q \Psi_Q$)
from $\langle extractFrame\ (!Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by** *simp+*
with $\langle \Psi \otimes \Psi_Q \triangleright R \mapsto_{\alpha} \prec R' \rangle \langle bn\ \alpha \#* P \rangle$ **have** $\Psi \triangleright R \parallel !P \mapsto_{\alpha} \prec (R' \parallel !P)$
by(*intro Par1*) (*assumption* | *simp*)
moreover from $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded\ P \rangle \langle guarded\ Q \rangle$ **have** $(\Psi, R' \parallel !P, R' \parallel !Q) \in Rel'$
by(*rule C1*)

ultimately show *?case by blast*
next
case(*cPar2* $Q' A_R \Psi_R$)
have $QTrans: \Psi \otimes \Psi_R \triangleright !Q \mapsto \alpha \prec Q'$ **and** $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
with $\langle bn \ \alpha \ \#* \ R \rangle \langle A_R \ \#* \ \alpha \rangle$ **have** $bn \ \alpha \ \#* \ \Psi_R$ **by**(*force dest: extractFrame-FreshChain*)
with $QTrans \langle (\Psi, P, Q) \in Rel \rangle \langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ Q \rangle \langle guarded \ P \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle$
obtain $P' S T$ **where** $PTrans: \Psi \otimes \Psi_R \triangleright !P \mapsto \alpha \prec P'$ **and** $(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$
and $suppT: ((supp \ T)::name \ set) \subseteq supp \ P'$ **and** $suppS: ((supp \ S)::name \ set) \subseteq supp \ Q'$
by(*drule-tac cSym*) (*auto dest: Der cExt*)
from $PTrans \ FrR \langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ P \rangle \langle A_R \ \#* \ \alpha \rangle \langle bn \ \alpha \ \#* \ R \rangle$ **have** $\Psi \triangleright R \parallel !P \mapsto \alpha \prec (R \parallel P')$
by(*intro Par2*) *auto*
moreover
{
from $\langle A_R \ \#* \ P \rangle \langle A_R \ \#* \ (!Q) \rangle \langle A_R \ \#* \ \alpha \rangle PTrans \ QTrans \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle distinct(bn \ \alpha) \rangle$ **have** $A_R \ \#* \ P'$ **and** $A_R \ \#* \ Q'$
by(*force dest: freeFreshChainDerivative*)
from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel \rangle FrR \langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ P' \rangle \langle A_R \ \#* \ P \rangle$ $suppT$ **have** $(\Psi, R \parallel P', R \parallel (T \parallel !P)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then **have** $(\Psi, R \parallel P', (R \parallel T) \parallel !P) \in Rel$ **by**(*blast intro: Assoc Trans*)
moreover **from** $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded \ P \rangle \langle guarded \ Q \rangle$ **have** $(\Psi, (R \parallel T) \parallel !P, (R \parallel T) \parallel !Q) \in Rel'$
by(*rule C1*)
moreover **from** $\langle (\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel \rangle \langle (\Psi \otimes \Psi_R, S, T) \in Rel \rangle$
have $(\Psi \otimes \Psi_R, Q', T \parallel !Q) \in Rel$
by(*blast intro: ParPres Trans*)
with $FrR \langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ P' \rangle \langle A_R \ \#* \ Q' \rangle \langle A_R \ \#* \ (!Q) \rangle$ $suppT \ suppS$ **have** $(\Psi, R \parallel Q', R \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then **have** $(\Psi, R \parallel Q', (R \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro: Assoc Trans*)
ultimately **have** $(\Psi, R \parallel P', R \parallel Q') \in Rel'$ **by**(*blast intro: cSym Compose*)
}
ultimately show *?case by blast*
next
case(*cComm1* $\Psi_Q M N R' A_R \Psi_R K xvec \ Q' A_Q$)
from $\langle extractFrame \ (!Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by** *simp+*
have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto M(|N|) \prec R'$ **and** $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
moreover **have** $QTrans: \Psi \otimes \Psi_R \triangleright !Q \mapsto K(|\nu * xvec|)(N) \prec Q'$ **by** *fact*
from $FrR \langle xvec \ \#* \ R \rangle \langle A_R \ \#* \ xvec \rangle$ **have** $xvec \ \#* \ \Psi_R$ **by**(*force dest: extract-FrameFreshChain*)

with $QTrans \langle (\Psi, P, Q) \in Rel \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* (!Q) \rangle \langle guarded P \rangle \langle xvec \#* K \rangle$
obtain $P' S T$ **where** $PTrans: \Psi \otimes \Psi_R \triangleright !P \mapsto K(\nu*xvec)\langle N \rangle \prec P'$ **and** $(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$
and $suppT: ((supp T)::name set) \subseteq supp P'$ **and** $suppS: ((supp S)::name set) \subseteq supp Q'$
apply(*drule-tac cSym*)
by (*metis Der bn.simps(3) cExt freshCompChain(1) optionFreshChain(1) psiFreshVec(7) subject.simps(3)*)
note $\langle \Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$
ultimately have $\Psi \triangleright R \parallel !P \mapsto \tau \prec (\nu*xvec)\langle R' \parallel P' \rangle$
using $PTrans \langle \Psi_Q = SBottom' \rangle \langle xvec \#* R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_R \#* P \rangle$
by(*intro Comm1*) (*assumption | simp*)+

moreover from $\langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle \langle A_R \#* xvec \rangle PTrans QTrans \langle xvec \#* K \rangle \langle distinct xvec \rangle$
have $A_R \#* P'$ **and** $A_R \#* Q'$ **by**(*force dest: outputFreshChainDerivative*)+
moreover with $RTrans FrR \langle distinct A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* N \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle \langle A_R \#* M \rangle$
obtain $\Psi' A_R' \Psi_R'$ **where** $FrR': extractFrame R' = \langle A_R', \Psi_R' \rangle$ **and** $\Psi_R \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \#* \Psi$
and $A_R' \#* P'$ **and** $A_R' \#* Q'$ **and** $A_R' \#* P$ **and** $A_R' \#* Q$
by(*auto intro: expandFrame[where C=(\Psi, P, P', Q, Q') and C'=\Psi]*)

moreover
{
from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel \rangle$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', T \parallel !P) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', P', T \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_R' \#* \Psi \rangle \langle A_R' \#* P' \rangle \langle A_R' \#* P \rangle$ $suppT$ **have** $(\Psi, R' \parallel P', R' \parallel (T \parallel !P)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then have $(\Psi, R' \parallel P', (R' \parallel T) \parallel !P) \in Rel$ **by**(*blast intro: Assoc Trans*)
with $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$ **have** $(\Psi, (\nu*xvec)\langle R' \parallel P' \rangle, ((\nu*xvec)\langle R' \parallel T \rangle) \parallel !P) \in Rel$
by(*metis ResPres psiFreshVec ScopeExt Trans*)
moreover from $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded P \rangle \langle guarded Q \rangle$ **have** $(\Psi, ((\nu*xvec)\langle R' \parallel T \rangle) \parallel !P, ((\nu*xvec)\langle R' \parallel T \rangle) \parallel !Q) \in Rel'$
by(*rule C1*)
moreover from $\langle (\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel \rangle \langle (\Psi \otimes \Psi_R, S, T) \in Rel \rangle$
have $(\Psi \otimes \Psi_R, Q', T \parallel !Q) \in Rel$
by(*blast intro: ParPres Trans*)
then have $((\Psi \otimes \Psi_R) \otimes \Psi', Q', T \parallel !Q) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', Q', T \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_R' \#* \Psi \rangle \langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* Q \rangle$ $suppT$ $suppS$ **have**

$(\Psi, R' \parallel Q', R' \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then have $(\Psi, R' \parallel Q', (R' \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro: Assoc Trans*)
with $\langle xvec \#* \Psi \rangle \langle xvec \#* (!Q) \rangle$ **have** $(\Psi, (\nu*xvec)(R' \parallel Q'), ((\nu*xvec)(R' \parallel T)) \parallel !Q) \in Rel$
by(*metis ResPres psiFreshVec ScopeExt Trans*)
ultimately have $(\Psi, (\nu*xvec)(R' \parallel P'), (\nu*xvec)(R' \parallel Q')) \in Rel'$ **by**(*blast intro: cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cComm2* $\Psi_Q M xvec N R' A_R \Psi_R K Q' A_Q$)
from $\langle extractFrame (!Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by**
simp+
have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto M(\nu*xvec)\langle N \rangle \prec R'$ **and** $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
then obtain $p \Psi' A_R' \Psi_R'$ **where** $S: set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$
and $FrR': extractFrame R' = \langle A_R', \Psi_R' \rangle$ **and** $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** $A_R' \#* \Psi$
and $A_R' \#* N$ **and** $A_R' \#* R'$ **and** $A_R' \#* P$ **and** $A_R' \#* Q$ **and** $(p \cdot xvec) \#* \Psi$
and $(p \cdot xvec) \#* P$ **and** $(p \cdot xvec) \#* Q$ **and** $xvec \#* A_R'$ **and** $(p \cdot xvec) \#* A_R'$
and *distinctPerm* p **and** $(p \cdot xvec) \#* R'$ **and** $(p \cdot xvec) \#* N$
using $\langle distinct\ A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_R \#* xvec \rangle \langle A_R \#* N \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle$
 $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* (!Q) \rangle \langle xvec \#* R \rangle \langle xvec \#* M \rangle \langle distinct\ xvec \rangle$
by(*auto intro: expandFrame[where C=(Ψ, P, Q) and C'=(Ψ, P, Q)]*)

from $RTrans\ S\ \langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* R' \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright R \mapsto M(\nu*(p \cdot xvec))\langle (p \cdot N) \rangle \prec (p \cdot R')$
apply(*simp add: residualInject*)
by(*subst boundOutputChainAlpha''[symmetric]*) *auto*

moreover have $QTrans: \Psi \otimes \Psi_R \triangleright !Q \mapsto K\langle N \rangle \prec Q'$ **by** *fact*
with $QTrans\ S\ \langle (p \cdot xvec) \#* N \rangle$ **have** $\Psi \otimes \Psi_R \triangleright !Q \mapsto K\langle (p \cdot N) \rangle \prec (p \cdot Q')$ **using** $\langle distinctPerm\ p \rangle \langle xvec \#* (!Q) \rangle \langle (p \cdot xvec) \#* Q \rangle$
by(*intro inputAlpha*) *auto*
with $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded\ P \rangle$
obtain $P' S T$ **where** $PTrans: \Psi \otimes \Psi_R \triangleright !P \mapsto K\langle (p \cdot N) \rangle \prec P'$ **and** $(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, (p \cdot Q'), S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$
and $suppT: ((supp\ T)::name\ set) \subseteq supp\ P'$ **and** $suppS: ((supp\ S)::name\ set) \subseteq supp(p \cdot Q')$
by(*drule-tac cSym*) (*auto dest: Der cExt*)
note $\langle \Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$
ultimately have $\Psi \triangleright R \parallel !P \mapsto \tau \prec (\nu*(p \cdot xvec))\langle (p \cdot R') \parallel P' \rangle$
using $PTrans\ FrR\ \langle \Psi_Q = SBottom' \rangle \langle (p \cdot xvec) \#* P \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_R \#* P \rangle$

by(*intro Comm2*) (*assumption* | *simp*)+

moreover from $\langle A_R' \#* P \rangle \langle A_R' \#* Q \rangle \langle A_R' \#* N \rangle S \langle xvec \#* A_R' \rangle \langle (p \cdot xvec) \#* A_R' \rangle$ *PTrans* *QTrans* $\langle distinctPerm p \rangle$ **have** $A_R' \#* P'$ **and** $A_R' \#* Q'$

apply(*drule-tac inputFreshChainDerivative*)

apply *simp*

apply(*subst pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst, symmetric, of - - p*], *simp*)

apply *fastforce*

using *QTrans* $\langle A_R' \#* N \rangle \langle A_R' \#* Q \rangle$ *inputFreshChainDerivative* **by force**

from $\langle xvec \#* P \rangle \langle (p \cdot xvec) \#* N \rangle$ *PTrans* $\langle distinctPerm p \rangle$ **have** $(p \cdot xvec) \#* (p \cdot P')$

apply(*drule-tac inputFreshChainDerivative*)

apply *simp*

by(*subst pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst, symmetric, of - - p*], *simp*)+

{

from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel \rangle$ **have** $(p \cdot (\Psi \otimes \Psi_R), (p \cdot P'), p \cdot (T \parallel !P)) \in Rel$

by(*rule Closed*)

with $\langle xvec \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi \rangle \langle xvec \#* P \rangle \langle (p \cdot xvec) \#* P \rangle S$ **have** $(\Psi \otimes (p \cdot \Psi_R), p \cdot P', (p \cdot T) \parallel !P) \in Rel$

by(*simp add: eqvts*)

then have $(\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', p \cdot P', (p \cdot T) \parallel !P) \in Rel$ **by**(*rule cExt*)

with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, (p \cdot P'), (p \cdot T) \parallel !P) \in Rel$

by(*metis Associativity StatEq compositionSym*)

with $FrR' \langle A_R' \#* \Psi \rangle \langle A_R' \#* P' \rangle \langle A_R' \#* P \rangle \langle xvec \#* A_R' \rangle \langle (p \cdot xvec) \#* A_R' \rangle S \langle distinctPerm p \rangle$ *suppT*

have $(\Psi, R' \parallel (p \cdot P'), R' \parallel ((p \cdot T) \parallel !P)) \in Rel$

apply(*intro FrameParPres*)

apply(*assumption* | *simp add: freshChainSimps*)+

by(*auto simp add: fresh-star-def fresh-def*)

then have $(\Psi, R' \parallel (p \cdot P'), (R' \parallel (p \cdot T)) \parallel !P) \in Rel$ **by**(*blast intro: Assoc Trans*)

with $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$ **have** $(\Psi, (\nu*xvec)(R' \parallel (p \cdot P')), ((\nu*xvec)(R' \parallel (p \cdot T))) \parallel !P) \in Rel$

by(*metis ResPres psiFreshVec ScopeExt Trans*)

then have $(\Psi, (\nu*(p \cdot xvec))((p \cdot R') \parallel P'), ((\nu*xvec)(R' \parallel (p \cdot T))) \parallel !P) \in Rel$

using $\langle (p \cdot xvec) \#* R' \rangle \langle (p \cdot xvec) \#* (p \cdot P') \rangle S \langle distinctPerm p \rangle$

apply(*erule-tac rev-mp*)

by(*subst resChainAlpha*[*of p*]) *auto*

moreover from $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded P \rangle \langle guarded Q \rangle$ **have** $(\Psi, ((\nu*xvec)(R' \parallel (p \cdot T))) \parallel !P, ((\nu*xvec)(R' \parallel (p \cdot T))) \parallel !Q) \in Rel'$

by(*rule C1*)

moreover from $\langle (\Psi \otimes \Psi_R, (p \cdot Q'), S \parallel !Q) \in Rel \rangle \langle (\Psi \otimes \Psi_R, S, T) \in Rel \rangle$ **have** $(\Psi \otimes \Psi_R, (p \cdot Q'), T \parallel !Q) \in Rel$

by(*blast intro: ParPres Trans*)

then have $(p \cdot (\Psi \otimes \Psi_R), p \cdot p \cdot Q', p \cdot (T \parallel !Q)) \in Rel$ **by** *(rule Closed)*
with $S \langle xvec \ \#* \ \Psi \rangle \langle (p \cdot xvec) \ \#* \ \Psi \rangle \langle xvec \ \#* \ (!Q) \rangle \langle (p \cdot xvec) \ \#* \ Q \rangle$
(distinctPerm p)
have $(\Psi \otimes (p \cdot \Psi_R), Q', (p \cdot T) \parallel !Q) \in Rel$ **by** *(simp add: eqvts)*
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', Q', (p \cdot T) \parallel !Q) \in Rel$ **by** *(rule cExt)*
with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, Q', (p \cdot T) \parallel !Q) \in Rel$
by *(metis Associativity StatEq compositionSym)*
with $FrR' \langle A_{R'} \ \#* \ \Psi \rangle \langle A_{R'} \ \#* \ P' \rangle \langle A_{R'} \ \#* \ Q' \rangle \langle A_{R'} \ \#* \ Q \rangle$ $suppT \ suppS \langle xvec$
 $\ \#* \ A_{R'} \rangle \langle (p \cdot xvec) \ \#* \ A_{R'} \rangle$ S *(distinctPerm p)*
have $(\Psi, R' \parallel Q', R' \parallel ((p \cdot T) \parallel !Q)) \in Rel$
apply *(intro FrameParPres)*
apply *(assumption | simp)+*
apply *(simp add: freshChainSimps)*
by *(auto simp add: fresh-star-def fresh-def)*
then have $(\Psi, R' \parallel Q', (R' \parallel (p \cdot T)) \parallel !Q) \in Rel$ **by** *(blast intro: Assoc*
Trans)
with $\langle xvec \ \#* \ \Psi \rangle \langle xvec \ \#* \ (!Q) \rangle$ **have** $(\Psi, (\nu * xvec)(R' \parallel Q'), ((\nu * xvec)(R' \parallel$
 $(p \cdot T))) \parallel !Q) \in Rel$
by *(metis ResPres psiFreshVec ScopeExt Trans)*
ultimately have $(\Psi, (\nu * (p \cdot xvec))((p \cdot R') \parallel P'), (\nu * xvec)(R' \parallel Q')) \in Rel'$
by *(blast intro: cSym Compose)*
}
ultimately show *?case by blast*
next
case *(cBrMerge $\Psi_Q \ M \ N \ R' \ A_R \ \Psi_R \ Q' \ A_Q$)*
from $\langle extractFrame \ (!Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by**
simp+
have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto_{iM} \langle N \rangle \prec R'$ **and** $FrR: extractFrame \ R = \langle A_R,$
 $\Psi_R \rangle$ **by** *fact+*
have $QTrans: \Psi \otimes \Psi_R \triangleright !Q \mapsto_{iM} \langle N \rangle \prec Q'$ **by** *fact*
with $\langle (\Psi, P, Q) \in Rel \rangle$ *(guarded P)*
obtain $P' \ S \ T$ **where** $PTrans: \Psi \otimes \Psi_R \triangleright !P \mapsto_{iM} \langle N \rangle \prec P'$ **and** $(\Psi \otimes \Psi_R,$
 $P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$
and $suppT: ((supp \ T)::name \ set) \subseteq supp \ P'$ **and** $suppS: ((supp \ S)::name \ set)$
 $\subseteq supp \ Q'$
by *(drule-tac cSym) (auto dest: Der intro: cExt)*
with $RTrans \ QTrans \ FrR$ **have** $\Psi \triangleright R \parallel !P \mapsto_{iM} \langle N \rangle \prec (R' \parallel P')$
using $PTrans \langle \Psi_Q = SBottom' \rangle \langle A_R \ \#* \ \Psi \rangle \langle A_R \ \#* \ R \rangle \langle A_R \ \#* \ M \rangle \langle A_R \ \#* \ P \rangle$
by *(intro BrMerge) (assumption | simp)+*

moreover from $\langle A_R \ \#* \ P \rangle \langle A_R \ \#* \ (!Q) \rangle \langle A_R \ \#* \ N \rangle$ $PTrans \ QTrans$
have $A_R \ \#* \ P'$ **and** $A_R \ \#* \ Q'$ **by** *(force dest: brinputFreshChainDerivative)+*
moreover with $RTrans \ FrR$ *(distinct A_R)* $\langle A_R \ \#* \ R \rangle \langle A_R \ \#* \ N \rangle \langle A_R \ \#* \ \Psi \rangle$
 $\langle A_R \ \#* \ P \rangle \langle A_R \ \#* \ (!Q) \rangle \langle A_R \ \#* \ M \rangle$
obtain $\Psi' \ A_{R'} \ \Psi_{R'}$ **where** $FrR': extractFrame \ R' = \langle A_{R'}, \Psi_{R'} \rangle$ **and** $\Psi_R \otimes \Psi'$
 $\simeq \Psi_{R'}$ **and** $A_{R'} \ \#* \ \Psi$
and $A_{R'} \ \#* \ P'$ **and** $A_{R'} \ \#* \ Q'$ **and** $A_{R'} \ \#* \ P$ **and** $A_{R'} \ \#* \ Q$
by *(auto intro: expandFrame[where $C=(\Psi, P, P', Q, Q')$ and $C'=\Psi$])*

moreover
{
from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel \rangle$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', T \parallel !P) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, P', T \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_{R'} \#* \Psi \rangle \langle A_{R'} \#* P' \rangle \langle A_{R'} \#* P \rangle$ *suppT* **have** $(\Psi, R' \parallel P', R' \parallel (T \parallel !P)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then have one: $(\Psi, R' \parallel P', (R' \parallel T) \parallel !P) \in Rel$ **by**(*blast intro: Assoc Trans*)
from $\langle (\Psi, P, Q) \in Rel \rangle \langle \text{guarded } P \rangle \langle \text{guarded } Q \rangle$ **have two:** $(\Psi, (R' \parallel T) \parallel !P, (R' \parallel T) \parallel !Q) \in Rel'$
by(*rule C1*)
from $\langle (\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel \rangle \langle (\Psi \otimes \Psi_R, S, T) \in Rel \rangle$ **have** $(\Psi \otimes \Psi_R, Q', T \parallel !Q) \in Rel$
by(*blast intro: ParPres Trans*)
then have $((\Psi \otimes \Psi_R) \otimes \Psi', Q', T \parallel !Q) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, Q', T \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_{R'} \#* \Psi \rangle \langle A_{R'} \#* P' \rangle \langle A_{R'} \#* Q' \rangle \langle A_{R'} \#* Q \rangle$ *suppT* *suppS* **have** $(\Psi, R' \parallel Q', R' \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres*) (*auto simp add: fresh-star-def fresh-def psi.supp*)
then have three: $(\Psi, R' \parallel Q', (R' \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro: Assoc Trans*)
from one two three **have** $(\Psi, (R' \parallel P'), (R' \parallel Q')) \in Rel'$ **by**(*blast intro: cSym Compose*)
}
ultimately show *?case* **by** *blast*
next
case(*cBrComm1* $\Psi_Q M N R' A_R \Psi_R \text{vec } Q' A_Q$)
from $\langle \downarrow M(\downarrow \nu * \text{vec}) \langle N \rangle = \alpha \rangle$ **have** *vec=bn* α
by(*auto simp add: action.inject*)
from $\langle \text{vec} = \text{bn } \alpha \rangle \langle \text{bn } \alpha \#* P \rangle \langle \text{bn } \alpha \#* R \rangle$
have *vec* $\#* P$ **and** *vec* $\#* R$ **by** *simp+*

from $\langle \text{extractFrame } (!Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = S\text{Bottom}'$ **by** *simp+*
have *RTrans:* $\Psi \otimes \Psi_Q \triangleright R \mapsto \downarrow M(\downarrow N) \prec R'$ **and** *FrR:* $\text{extractFrame } R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
moreover have *QTrans:* $\Psi \otimes \Psi_R \triangleright !Q \mapsto \downarrow M(\downarrow \nu * \text{vec}) \langle N \rangle \prec Q'$ **by** *fact*
from *FrR* $\langle \text{vec } \#* R \rangle \langle A_R \#* \text{vec} \rangle$ **have** *vec* $\#* \Psi_R$ **by**(*force dest: extractFrameFreshChain*)
with *QTrans* $\langle (\Psi, P, Q) \in Rel \rangle \langle \text{vec } \#* \Psi \rangle \langle \text{vec } \#* P \rangle \langle \text{vec } \#* (!Q) \rangle \langle \text{guarded } P \rangle \langle \text{vec } \#* M \rangle$
obtain $P' S T$ **where** *PTrans:* $\Psi \otimes \Psi_R \triangleright !P \mapsto \downarrow M(\downarrow \nu * \text{vec}) \langle N \rangle \prec P'$ **and** $(\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$

and $\text{supp}T: ((\text{supp } T)::\text{name set}) \subseteq \text{supp } P' \text{ and } \text{supp}S: ((\text{supp } S)::\text{name set})$
 $\subseteq \text{supp } Q'$
apply(*drule-tac cSym*)
by(*metis Der ⟨bn α #* Q⟩ ⟨xvec = bn α⟩ cBrComm1.hyps(30) cExt cSim.hyps(6)*
freshCompChain(1))

ultimately have $\Psi \triangleright R \parallel !P \mapsto_i M(\nu * xvec)(N) \prec (R' \parallel P')$
using $PTrans \langle \Psi_Q = SBottom' \rangle \langle xvec \#* R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_R \#* P \rangle$
by(*intro BrComm1 (assumption | simp)+*)

moreover from $\langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle \langle A_R \#* xvec \rangle PTrans QTrans \langle xvec \#* M \rangle \langle distinct xvec \rangle$
have $A_R \#* P'$ **and** $A_R \#* Q'$ **by**(*force dest: broutputFreshChainDerivative*)
moreover with $RTrans FrR \langle distinct A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* N \rangle \langle A_R \#* \Psi \rangle$
 $\langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle \langle A_R \#* M \rangle$
obtain $\Psi' A_R' \Psi_R'$ **where** $FrR': \text{extractFrame } R' = \langle A_R', \Psi_R' \rangle$ **and** $\Psi_R \otimes \Psi' \simeq \Psi_R' \text{ and } A_R' \#* \Psi$
and $A_R' \#* P'$ **and** $A_R' \#* Q'$ **and** $A_R' \#* P$ **and** $A_R' \#* Q$
by(*auto intro: expandFrame[where C=(Ψ, P, P', Q, Q') and C'=Ψ]*)

moreover
{
from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in Rel \rangle$ **have** $((\Psi \otimes \Psi_R) \otimes \Psi', P', T \parallel !P) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', P', T \parallel !P) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_R' \#* \Psi \rangle \langle A_R' \#* P' \rangle \langle A_R' \#* P \rangle$ **supp** T **have** $(\Psi, R' \parallel P', R' \parallel (T \parallel !P)) \in Rel$
by(*intro FrameParPres (auto simp add: fresh-star-def fresh-def psi.supp)*)
then have one: $(\Psi, R' \parallel P', (R' \parallel T) \parallel !P) \in Rel$ **by**(*blast intro: Assoc Trans*)
from $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded P \rangle \langle guarded Q \rangle$ **have two:** $(\Psi, (R' \parallel T) \parallel !P, (R' \parallel T) \parallel !Q) \in Rel'$
by(*rule C1*)
from $\langle (\Psi \otimes \Psi_R, Q', S \parallel !Q) \in Rel \rangle \langle (\Psi \otimes \Psi_R, S, T) \in Rel \rangle$ **have** $(\Psi \otimes \Psi_R, Q', T \parallel !Q) \in Rel$
by(*blast intro: ParPres Trans*)
then have $((\Psi \otimes \Psi_R) \otimes \Psi', Q', T \parallel !Q) \in Rel$ **by**(*rule cExt*)
with $\langle \Psi_R \otimes \Psi' \simeq \Psi_R' \rangle$ **have** $(\Psi \otimes \Psi_R', Q', T \parallel !Q) \in Rel$
by(*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_R' \#* \Psi \rangle \langle A_R' \#* P' \rangle \langle A_R' \#* Q' \rangle \langle A_R' \#* Q \rangle$ **supp** T **supp** S **have** $(\Psi, R' \parallel Q', R' \parallel (T \parallel !Q)) \in Rel$
by(*intro FrameParPres (auto simp add: fresh-star-def fresh-def psi.supp)*)
then have three: $(\Psi, R' \parallel Q', (R' \parallel T) \parallel !Q) \in Rel$ **by**(*blast intro: Assoc Trans*)
from one two three have $(\Psi, (R' \parallel P'), (R' \parallel Q')) \in Rel'$ **by**(*blast intro: cSym Compose*)
}

ultimately show $?case$ **by** *blast*
next
case(*cBrComm2* Ψ_Q M *xvec* N R' A_R Ψ_R Q' A_Q)
from $\langle \downarrow M(\nu * xvec) \rangle \langle N \rangle = \alpha$ **have** $xvec = bn \alpha$
by(*auto simp add: action.inject*)
from $\langle xvec = bn \alpha \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* R \rangle$
have $xvec \#* P$ **and** $xvec \#* R$ **by** *simp+*

from $\langle extractFrame (!Q) \rangle = \langle A_Q, \Psi_Q \rangle$ **have** $A_Q = []$ **and** $\Psi_Q = SBottom'$ **by**
simp+
have $RTrans: \Psi \otimes \Psi_Q \triangleright R \mapsto \downarrow M(\nu * xvec) \langle N \rangle \prec R'$ **and** $FrR: extractFrame$
 $R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
then obtain p Ψ' A_R' Ψ_R' **where** $S: set\ p \subseteq set\ xvec \times set(p \cdot xvec)$
and $FrR': extractFrame\ R' = \langle A_R', \Psi_R' \rangle$ **and** $(p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_R'$ **and** A_R'
 $\#* \Psi$
and $A_R' \#* N$ **and** $A_R' \#* M$ **and** $A_R' \#* R$ **and** $A_R' \#* R'$ **and** $A_R' \#* P$
and $A_R' \#* Q$ **and** $(p \cdot xvec) \#* \Psi$
and $(p \cdot xvec) \#* P$ **and** $(p \cdot xvec) \#* Q$ **and** $xvec \#* A_R'$ **and** $(p \cdot xvec) \#*$
 A_R'
and *distinctPerm* p **and** $(p \cdot xvec) \#* R$ **and** $(p \cdot xvec) \#* R'$ **and** $(p \cdot xvec)$
 $\#* N$ **and** $(p \cdot xvec) \#* M$
using $\langle distinct\ A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle A_R \#* xvec \rangle \langle A_R \#* N \rangle \langle A_R \#*$
 $\Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* (!Q) \rangle$
 $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* (!Q) \rangle \langle xvec \#* R \rangle \langle xvec \#* M \rangle \langle distinct\ xvec \rangle$
by(*auto intro: expandFrame[where* $C=(\Psi, P, R, Q, M)$ **and** $C'=(\Psi, P, R,$
 $Q, M)$])

from $RTrans\ S$ $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* R' \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright R$
 $\mapsto \downarrow M(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec (p \cdot R')$
apply(*simp add: residualInject*)
by(*subst boundOutputChainAlpha''[symmetric]*) *auto*

moreover have $QTrans: \Psi \otimes \Psi_R \triangleright !Q \mapsto \downarrow M(!N) \prec Q'$ **by** *fact*
with $QTrans\ S$ $\langle (p \cdot xvec) \#* N \rangle$ **have** $\Psi \otimes \Psi_R \triangleright !Q \mapsto \downarrow M((p \cdot N)) \prec (p \cdot$
 $Q')$ **using** $\langle distinctPerm\ p \rangle \langle xvec \#* (!Q) \rangle \langle (p \cdot xvec) \#* Q \rangle$
by(*intro brinputAlpha*) *auto*
with $\langle (\Psi, P, Q) \in Rel \rangle \langle guarded\ P \rangle$
obtain P' S T **where** $PTrans: \Psi \otimes \Psi_R \triangleright !P \mapsto \downarrow M((p \cdot N)) \prec P'$ **and** $(\Psi$
 $\otimes \Psi_R, P', T \parallel !P) \in Rel$
and $(\Psi \otimes \Psi_R, (p \cdot Q'), S \parallel !Q) \in Rel$ **and** $(\Psi \otimes \Psi_R, S, T) \in Rel$
and $suppT: ((supp\ T)::name\ set) \subseteq supp\ P'$ **and** $suppS: ((supp\ S)::name\ set)$
 $\subseteq supp(p \cdot Q')$
by(*drule-tac cSym*) (*auto dest: Der cExt*)
ultimately have $\Psi \triangleright R \parallel !P \mapsto \downarrow M(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec ((p \cdot R') \parallel P')$
using $PTrans\ FrR\ \langle \Psi_Q = SBottom' \rangle \langle (p \cdot xvec) \#* P \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* R \rangle$
 $\langle A_R \#* M \rangle \langle A_R \#* P \rangle$
by(*intro BrComm2*) (*assumption | simp*)
then have $p \cdot (\Psi \triangleright R \parallel !P \mapsto \downarrow M(\nu * (p \cdot xvec)) \langle (p \cdot N) \rangle \prec ((p \cdot R') \parallel P'))$
by *simp*

with $\langle \text{distinctPerm } p \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot R) \parallel !(p \cdot P) \mapsto_i (p \cdot M) \langle \nu * \text{vec} \rangle \langle N \rangle$
 $\prec (R' \parallel (p \cdot P'))$
by (*simp add: eqvts*)
with $S \langle \text{vec} \#* \Psi \rangle \langle (p \cdot \text{vec}) \#* \Psi \rangle \langle \text{vec} \#* R \rangle \langle (p \cdot \text{vec}) \#* R \rangle$
 $\langle \text{vec} \#* P \rangle \langle (p \cdot \text{vec}) \#* P \rangle \langle \text{vec} \#* M \rangle \langle (p \cdot \text{vec}) \#* M \rangle$
have $\Psi \triangleright R \parallel !P \mapsto_i M \langle \nu * \text{vec} \rangle \langle N \rangle \prec (R' \parallel (p \cdot P'))$
by *simp*

moreover from $\langle A_{R'} \#* P \rangle \langle A_{R'} \#* Q \rangle \langle A_{R'} \#* N \rangle S \langle \text{vec} \#* A_{R'} \rangle \langle (p \cdot \text{vec}) \#* A_{R'} \rangle$
 $P\text{Trans } Q\text{Trans } \langle \text{distinctPerm } p \rangle$ **have** $A_{R'} \#* P'$ **and** $A_{R'} \#* Q'$
apply (*drule-tac brinputFreshChainDerivative, simp*)
apply (*subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, symmetric, of - - p], simp*)
apply *force*
using $Q\text{Trans } \langle A_{R'} \#* N \rangle \langle A_{R'} \#* Q \rangle$ *brinputFreshChainDerivative* **by** *force*
from $\langle \text{vec} \#* P \rangle \langle (p \cdot \text{vec}) \#* N \rangle P\text{Trans } \langle \text{distinctPerm } p \rangle$ **have** $(p \cdot \text{vec}) \#* (p \cdot P')$
apply (*drule-tac brinputFreshChainDerivative, simp*)
apply (*subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, symmetric, of - - p], simp*)
by (*subst pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst, symmetric, of - - p], simp*)

{
from $\langle (\Psi \otimes \Psi_R, P', T \parallel !P) \in \text{Rel} \rangle$ **have** $(p \cdot (\Psi \otimes \Psi_R), (p \cdot P'), p \cdot (T \parallel !P)) \in \text{Rel}$
by (*rule Closed*)
with $\langle \text{vec} \#* \Psi \rangle \langle (p \cdot \text{vec}) \#* \Psi \rangle \langle \text{vec} \#* P \rangle \langle (p \cdot \text{vec}) \#* P \rangle S$ **have** $(\Psi \otimes (p \cdot \Psi_R), p \cdot P', (p \cdot T) \parallel !P) \in \text{Rel}$
by (*simp add: eqvts*)
then have $((\Psi \otimes (p \cdot \Psi_R)) \otimes \Psi', p \cdot P', (p \cdot T) \parallel !P) \in \text{Rel}$
by (*rule cExt*)
with $\langle (p \cdot \Psi_R) \otimes \Psi' \simeq \Psi_{R'} \rangle$ **have** $(\Psi \otimes \Psi_{R'}, (p \cdot P'), (p \cdot T) \parallel !P) \in \text{Rel}$
by (*metis Associativity StatEq compositionSym*)
with $FrR' \langle A_{R'} \#* \Psi \rangle \langle A_{R'} \#* P' \rangle \langle A_{R'} \#* P \rangle \langle \text{vec} \#* A_{R'} \rangle \langle (p \cdot \text{vec}) \#* A_{R'} \rangle S$
 $\langle \text{distinctPerm } p \rangle$ *suppT*
have $(\Psi, R' \parallel (p \cdot P'), R' \parallel ((p \cdot T) \parallel !P)) \in \text{Rel}$
apply (*intro FrameParPres*)
apply (*assumption | simp add: freshChainSimps*)
by (*auto simp add: fresh-star-def fresh-def*)
then have one: $(\Psi, R' \parallel (p \cdot P'), (R' \parallel (p \cdot T)) \parallel !P) \in \text{Rel}$ **by** (*blast intro: Assoc Trans*)
from $\langle (\Psi, P, Q) \in \text{Rel} \rangle \langle \text{guarded } P \rangle \langle \text{guarded } Q \rangle$ **have two:** $(\Psi, (R' \parallel (p \cdot T)) \parallel !P, (R' \parallel (p \cdot T)) \parallel !Q) \in \text{Rel}'$
by (*rule C1*)
from $\langle (\Psi \otimes \Psi_R, (p \cdot Q'), S \parallel !Q) \in \text{Rel} \rangle \langle (\Psi \otimes \Psi_R, S, T) \in \text{Rel} \rangle$ **have** $(\Psi \otimes \Psi_R, (p \cdot Q'), T \parallel !Q) \in \text{Rel}$
by (*blast intro: ParPres Trans*)
then have $(p \cdot (\Psi \otimes \Psi_R), p \cdot p \cdot Q', p \cdot (T \parallel !Q)) \in \text{Rel}$ **by** (*rule Closed*)

```

    with S ⟨xvec #* Ψ⟩ ⟨(p · xvec) #* Ψ⟩ ⟨xvec #* (!Q)⟩ ⟨(p · xvec) #* Q⟩
  ⟨distinctPerm p⟩
  have (Ψ ⊗ (p · ΨR), Q', (p · T) || !Q) ∈ Rel by(simp add: eqvts)
  then have ((Ψ ⊗ (p · ΨR)) ⊗ Ψ', Q', (p · T) || !Q) ∈ Rel by(rule cExt)
  with ⟨(p · ΨR) ⊗ Ψ' ≈ ΨR'⟩ have (Ψ ⊗ ΨR', Q', (p · T) || !Q) ∈ Rel
    by(metis Associativity StatEq compositionSym)
  with FrR' ⟨AR' #* Ψ⟩ ⟨AR' #* P'⟩ ⟨AR' #* Q'⟩ ⟨AR' #* Q⟩ suppT suppS ⟨xvec
#* AR'⟩ ⟨(p · xvec) #* AR'⟩ S ⟨distinctPerm p⟩
  have (Ψ, R' || Q', R' || ((p · T) || !Q)) ∈ Rel
    apply(intro FrameParPres)
    apply(assumption | simp)+
    apply(simp add: freshChainSimps)
    by(auto simp add: fresh-star-def fresh-def)
  then have three: (Ψ, R' || Q', (R' || (p · T)) || !Q) ∈ Rel by(blast intro: Assoc
Trans)
  from one two three have (Ψ, (R' || (p · P')), (R' || Q')) ∈ Rel' by(blast intro:
cSym Compose)
}
ultimately show ?case by blast
qed
qed

```

notation relcomp (infixr 0 75)

end

end

theory Sim-Struct-Cong

imports Simulation HOL-Library.Multiset

begin

This file is a (heavily modified) variant of the theory *Psi_Calculi.Sim_Struct_Cong* from [1].

lemma partitionListLeft:

```

  assumes xs@ys=xs'@y#ys'
  and y ∈ set xs
  and distinct(xs@ys)

```

obtains zs where xs = xs'@y#zs and ys'=zs@ys

```

  using assms
  by(force simp add: append-eq-append-conv2 append-eq-Cons-conv)

```

lemma partitionListRight:

```

  assumes xs@ys=xs'@y#ys'
  and y ∈ set ys
  and distinct(xs@ys)

```

obtains zs where xs' = xs@zs and ys=zs@y#ys'

```

using assms
by(force simp add: append-eq-append-conv2 append-eq-Cons-conv)

context env begin

lemma resOutputCases''''[consumes 8, case-names cOpen cRes]:
  fixes  $\Psi$  :: 'b
    and x :: name
    and zvec :: name list
    and P :: ('a, 'b, 'c) psi
    and  $\alpha$  :: 'a action
    and P' :: ('a, 'b, 'c) psi
    and C :: 'f::fs-name

assumes Trans:  $\Psi \triangleright (\nu x)P \mapsto \alpha \prec P'$ 
  and 1:  $x \# \Psi$ 
  and 2:  $x \# \alpha$ 
  and 3:  $x \# P'$ 
  and 4:  $bn \alpha \#* \Psi$ 
  and 5:  $bn \alpha \#* P$ 
  and 6:  $bn \alpha \#* \text{subject } \alpha$ 
  and  $\alpha = M(\nu * zvec)\langle N \rangle$ 
  and rOpen:  $\bigwedge M \text{ xvec yvec } y N P'. \llbracket \Psi \triangleright [(x, y)] \cdot P \rrbracket \mapsto M(\nu *(xvec @ yvec))\langle N \rangle$ 
 $\prec P'; y \in \text{supp } N;$ 
 $x \# N; x \# P'; x \neq y; y \# xvec; y \# yvec; y \# M;$ 
distinct xvec; distinct yvec;
 $xvec \#* \Psi; y \# \Psi; yvec \#* \Psi; xvec \#* P; y \# P;$ 
 $yvec \#* P; xvec \#* M; y \# M;$ 
 $yvec \#* M; xvec \#* yvec \rrbracket \implies$ 
 $\text{Prop } (M(\nu *(xvec @ y\#yvec))\langle N \rangle) P'$ 
  and rScope:  $\bigwedge P'. \llbracket \Psi \triangleright P \mapsto \alpha \prec P' \rrbracket \implies \text{Prop } \alpha ((\nu x)P')$ 

shows Prop  $\alpha P'$ 
proof –
  from Trans have distinct (bn  $\alpha$ ) by(auto dest: boundOutputDistinct)
  show ?thesis using Trans 1 2 3 4 5 6  $\langle \alpha = M(\nu * zvec)\langle N \rangle \rangle$  rOpen rScope
  proof(induct rule: resCases'[where  $C = (zvec, C)$ ])
    case cBrOpen
    then show ?case
    by(auto simp add: residualInject boundOutputApp)
  next
    case cRes
    then show ?case
    by(auto simp add: residualInject boundOutputApp)
  next
    case cBrClose
    then show ?case
    by(auto simp add: residualInject boundOutputApp)
  next

```

case(*cOpen* $M' \text{ zvec } y \ N' \ P'$)
show *?case*
using $\langle \Psi \triangleright [(x, y)] \cdot P \mapsto M'(\nu^*(\text{zvec} @ yvec)) \langle N' \rangle \prec P' \rangle \langle y \in \text{supp } N' \rangle$
 $\langle x \# N' \rangle \langle x \# P' \rangle$
 $\langle x \neq y \rangle \langle y \# \text{zvec} \rangle \langle y \# yvec \rangle \langle y \# M' \rangle \langle \text{distinct } \text{zvec} \rangle \langle \text{distinct } yvec \rangle \langle \text{zvec} \#*$
 $\Psi \rangle$
 $\langle y \# \Psi \rangle \langle yvec \#* \Psi \rangle \langle \text{zvec} \#* P \rangle \langle y \# P \rangle \langle yvec \#* P \rangle \langle \text{zvec} \#* M' \rangle \langle y \# M' \rangle$
 $\langle yvec \#* M' \rangle \langle \text{zvec} \#* yvec \rangle$
by(*rule cOpen(22)*)
qed
qed

lemma *resOutputCases''''*[*consumes* γ , *case-names* *cOpen cRes*]:

fixes Ψ :: 'b
and x :: *name*
and zvec :: *name list*
and P :: ('a, 'b, 'c) *psi*
and P' :: ('a, 'b, 'c) *psi*
and C :: 'f::*fs-name*

assumes *Trans*: $\Psi \triangleright (\nu x)P \mapsto M(\nu^* \text{zvec}) \langle N \rangle \prec P'$
and 1 : $x \# \Psi$
and $x \# M(\nu^* \text{zvec}) \langle N \rangle$
and 3 : $x \# P'$
and $\text{zvec} \#* \Psi$
and $\text{zvec} \#* P$
and $\text{zvec} \#* M$
and *rOpen*: $\bigwedge M' \text{ zvec } y \ N' \ P'. \llbracket \Psi \triangleright [(x, y)] \cdot P \mapsto M'(\nu^*(\text{zvec} @ yvec)) \langle N' \rangle \prec P'; y \in \text{supp } N' \rrbracket$
 $x \# N'; x \# P'; x \neq y; y \# \text{zvec}; y \# yvec; y \# M';$
distinct zvec; distinct yvec;
 $\text{zvec} \#* \Psi; y \# \Psi; yvec \#* \Psi; \text{zvec} \#* P; y \# P;$
 $yvec \#* P; \text{zvec} \#* M'; y \# M';$
 $yvec \#* M'; \text{zvec} \#* yvec; M'(\nu^*(\text{zvec} @ y\#yvec)) \langle N' \rangle$
 $= M(\nu^* \text{zvec}) \langle N \rangle \rrbracket \implies$
 $\text{Prop } P'$
and *rScope*: $\bigwedge P'. \llbracket \Psi \triangleright P \mapsto M(\nu^* \text{zvec}) \langle N \rangle \prec P' \rrbracket \implies \text{Prop } ((\nu x)P')$

shows *Prop P'*

proof –

from *Trans* **have** *distinct zvec* **by**(*auto dest: boundOutputDistinct*)
obtain al **where** $al = M(\nu^* \text{zvec}) \langle N \rangle$ **by** *simp*
from $\langle al = M(\nu^* \text{zvec}) \langle N \rangle \rangle$ *Trans* $\langle \text{zvec} \#* \Psi \rangle \langle \text{zvec} \#* P \rangle \langle \text{zvec} \#* M \rangle$
have α *Trans*: $\Psi \triangleright (\nu x)P \mapsto al \prec P'$ **and** 4 : $bn \ al \ \#* \ \Psi$ **and** 5 : $bn \ al \ \#* \ P$ **and**
 6 : $bn \ al \ \#* \ \text{subject } al$
by *simp+*
from $\langle x \# M(\nu^* \text{zvec}) \langle N \rangle \rangle \langle al = M(\nu^* \text{zvec}) \langle N \rangle \rangle$ **have** 2 : $x \# al$ **by** *simp*
show *?thesis* **using** α *Trans* $1 \ 2 \ 3 \ 4 \ 5 \ 6 \ \langle al = M(\nu^* \text{zvec}) \langle N \rangle \rangle$ *rOpen rScope*
proof(*induct rule: resCases'[where C=(zvec, C)]*)

```

    case cBrOpen
  then show ?case
    by(auto simp add: residualInject boundOutputApp)
  next
    case cBrClose
  then show ?case
    by(auto simp add: residualInject boundOutputApp)
  next
    case(cOpen M' xvec yvec y N' P')
  show ?case
    using ⟨Ψ ▷ [(x, y)] · P ⟶ M'(ν*(xvec @ yvec))⟨N'⟩ < P'⟩ ⟨y ∈ supp N'⟩
  ⟨x # N'⟩ ⟨x # P'⟩
    ⟨x ≠ y⟩ ⟨y # xvec⟩ ⟨y # yvec⟩ ⟨y # M'⟩ ⟨distinct xvec⟩ ⟨distinct yvec⟩ ⟨xvec #*
  Ψ⟩
    ⟨y # Ψ⟩ ⟨yvec #* Ψ⟩ ⟨xvec #* P⟩ ⟨y # P⟩ ⟨yvec #* P⟩ ⟨xvec #* M'⟩ ⟨y # M'⟩
    ⟨yvec #* M'⟩ ⟨xvec #* yvec⟩ ⟨M'(ν*(xvec @ y # yvec))⟨N'⟩ = M(ν*zvec)⟨N⟩⟩
    by(rule cOpen(22))
  next
    case (cRes P')
  from ⟨Ψ ▷ P ⟶ al < P'⟩ ⟨al = M(ν*zvec)⟨N⟩⟩
  show ?case
    by (simp add: cRes(4))
qed
qed

```

lemma *resBrOutputCases'*[*consumes* γ , *case-names* *cBrOpen* *cRes*]:

```

  fixes Ψ    :: 'b
  and x      :: name
  and zvec   :: name list
  and P      :: ('a, 'b, 'c) psi
  and P'     :: ('a, 'b, 'c) psi
  and C      :: 'f::fs-name

```

```

assumes Trans: Ψ ▷ (νx)P ⟶iM(ν*zvec)⟨N⟩ < P'
and 1: x # Ψ
and x #iM(ν*zvec)⟨N⟩
and 3: x # P'
and zvec #* Ψ
and zvec #* P
and zvec #* M
and rBrOpen: ∧M' xvec yvec y N' P'. [Ψ ▷ ((x, y) · P) ⟶iM'(ν*(xvec@yvec))⟨N'⟩
< P'; y ∈ supp N'];
    x # N'; x # P'; x ≠ y; y # xvec; y # yvec; y # M';
distinct xvec; distinct yvec;
    xvec #* Ψ; y # Ψ; yvec #* Ψ; xvec #* P; y # P;
yvec #* P; xvec #* M'; y # M';
    yvec #* M'; xvec #* yvec; iM'(ν*(xvec@y#yvec))⟨N'⟩
= iM(ν*zvec)⟨N⟩] ⟹
    Prop P'

```

and $rScope: \bigwedge P'. [\Psi \triangleright P \mapsto_{iM} (\nu * zvec) \langle N \rangle \prec P'] \implies Prop ((\nu x) P')$

shows $Prop P'$

proof –

from $Trans$ have $distinct\ zvec$ by $(auto\ dest: boundOutputDistinct)$

obtain al where $al =_{iM} (\nu * zvec) \langle N \rangle$ by $simp$

from $\langle al =_{iM} (\nu * zvec) \langle N \rangle \rangle Trans\ \langle zvec \#* \Psi \rangle \langle zvec \#* P \rangle \langle zvec \#* M \rangle$

have $\alpha Trans: \Psi \triangleright (\nu x) P \mapsto al \prec P'$ and $4: bn\ al\ \#* \Psi$ and $5: bn\ al\ \#* P$ and $6: bn\ al\ \#*$ subject al

by $simp+$

from $\langle x \#_{iM} (\nu * zvec) \langle N \rangle \rangle \langle al =_{iM} (\nu * zvec) \langle N \rangle \rangle$ have $2: x \# al$ by $simp$

show $?thesis$ using $\alpha Trans\ 1\ 2\ 3\ 4\ 5\ 6\ \langle al =_{iM} (\nu * zvec) \langle N \rangle \rangle\ rBrOpen\ rScope$

proof (induct rule: $resCases'$ [where $C = (zvec, C)$])

case $cBrOpen$

then show $?case$

by $(auto\ simp\ add: residualInject\ boundOutputApp)$

next

case $cBrClose$

then show $?case$

by $(auto\ simp\ add: residualInject\ boundOutputApp)$

next

case $(cOpen\ M'\ xvec\ yvec\ y\ N'\ P')$

then show $?case$

by $(auto\ simp\ add: residualInject\ boundOutputApp)$

next

case $(cRes\ P')$

from $\langle \Psi \triangleright P \mapsto al \prec P' \rangle \langle al =_{iM} (\nu * zvec) \langle N \rangle \rangle$

show $?case$

by $(simp\ add: cRes(4))$

qed

qed

lemma $brOutputFreshSubject$:

fixes $x::name$

assumes $\Psi \triangleright P \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec P'$

and $xvec \#* M$

and $x \# P$

shows $x \# M$

using $assms$

proof (nominal-induct avoiding: x rule: $brOutputInduct'$)

case $(cAlpha\ \Psi\ P\ M\ xvec\ N\ P'\ p)$

then show $?case$ by $simp$

next

case $(cBrOutput\ \Psi\ M\ K\ N\ P)$

then show $?case$

by $(auto\ simp\ add: fresh-def\ psi.supp\ dest: chanOutConSupp)$

next

case $(cCase\ \Psi\ P\ M\ xvec\ N\ P'\ \varphi\ Cs)$ then show $?case$

by $(induct\ Cs)\ auto$

```

next
  case(cPar1  $\Psi \Psi_Q P M \text{vec } N P' A_Q Q$ ) then show ?case
    by simp
next
  case cPar2 then show ?case by simp
next
  case cBrComm1 then show ?case by simp
next
  case cBrComm2 then show ?case by simp
next
  case cBrOpen then show ?case by (simp add: fresh-abs-fun-iff[OF pt-name-inst,
    OF at-name-inst, OF fin-supp])
next
  case cScope then show ?case by (simp add: fresh-abs-fun-iff[OF pt-name-inst,
    OF at-name-inst, OF fin-supp])
next
  case cBang then show ?case by simp
qed

```

lemma *brInputFreshSubject*:

```

fixes  $x :: \text{name}$ 
assumes  $\Psi \triangleright P \mapsto \iota M(|N|) \prec P'$ 
  and  $x \# P$ 
shows  $x \# M$ 
using assms
proof(nominal-induct avoiding:  $x$  rule: brInputInduct)
  case(cBrInput  $\Psi K M \text{vec } N Tvec P y$ )
  then show ?case
    by(auto simp add: fresh-def psi.supp dest: chanInConSupp)
next
  case(cCase  $\Psi P M N P' \varphi Cs y$ ) then show ?case
    by(induct  $Cs$ ) auto
next
  case(cPar1  $\Psi \Psi_Q P M N P' A_Q Q y$ ) then show ?case
    by simp
next
  case cPar2 then show ?case by simp
next
  case cBrMerge then show ?case by simp
next
  case cScope then show ?case by (simp add: fresh-abs-fun-iff[OF pt-name-inst,
    OF at-name-inst, OF fin-supp])
next
  case cBang then show ?case by simp
qed

```

lemma *resComm*:

```

fixes  $\Psi :: 'b$ 
  and  $x :: \text{name}$ 

```



```

and  $y :: name$ 
and  $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 
and  $P :: ('a, 'b, 'c) psi$ 

assumes  $x \# \Psi$ 
and  $y \# \Psi$ 
and  $eqvt Rel$ 
and  $R1: \bigwedge \Psi' Q. (\Psi', Q, Q) \in Rel$ 
and  $R2: \bigwedge \Psi' a b Q. \llbracket a \# \Psi'; b \# \Psi' \rrbracket \implies (\Psi', (\nu a)((\nu b)Q), (\nu b)((\nu a)Q)) \in Rel$ 
and  $R3: \bigwedge \Psi' xvec yvec Q. \llbracket xvec \#* \Psi'; mset xvec = mset yvec \rrbracket \implies (\Psi', (\nu *xvec)Q, (\nu *yvec)Q) \in Rel$ 

shows  $\Psi \triangleright (\nu x)((\nu y)P) \rightsquigarrow[Rel] (\nu y)((\nu x)P)$ 
proof(cases  $x=y$ )
  assume  $x = y$ 
  then show ?thesis using  $R1$ 
    by(force intro: reflexive)
  next
    assume  $x \neq y$ 
    note  $\langle eqvt Rel \rangle$ 
    moreover from  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$  have  $[x, y] \#* \Psi$  by(simp add: fresh-star-def)
    moreover have  $[x, y] \#* (\nu x)((\nu y)P)$  by(simp add: abs-fresh)
    moreover have  $[x, y] \#* (\nu y)((\nu x)P)$  by(simp add: abs-fresh)
    ultimately show ?thesis
    proof(induct rule: simIChainFresh[where  $C=(x, y)$ ])
      case(cSim  $\alpha P'$ )
        from  $\langle bn \alpha \#* (x, y) \rangle \langle bn \alpha \#* ((\nu x)((\nu y)P)) \rangle$  have  $x \# bn \alpha$  and  $y \# bn \alpha$ 
    and  $bn \alpha \#* P$  by simp+
      from  $\langle [x, y] \#* \alpha \rangle$  have  $x \# \alpha$  and  $y \# \alpha$  by simp+
      from  $\langle [x, y] \#* P' \rangle$  have  $x \# P'$  and  $y \# P'$  by simp+
      from  $\langle bn \alpha \#* P \rangle \langle x \# \alpha \rangle$  have  $bn \alpha \#* (\nu x)P$  by(simp add: abs-fresh)
      with  $\langle \Psi \triangleright (\nu y)((\nu x)P) \mapsto \alpha \prec P' \rangle \langle y \# \Psi \rangle \langle y \# \alpha \rangle \langle y \# P' \rangle \langle bn \alpha \#* \Psi \rangle$ 
      show ?case using  $\langle bn \alpha \#* subject \alpha \rangle \langle x \# \alpha \rangle \langle x \# P' \rangle \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* subject \alpha \rangle \langle y \# \alpha \rangle$ 
      proof(induct rule: resCases'[where  $C=x$ ])
        case(cOpen  $M yvec1 yvec2 y' N P'$ )
          from  $\langle yvec1 \#* yvec2 \rangle \langle distinct yvec1 \rangle \langle distinct yvec2 \rangle$  have  $distinct(yvec1 @ yvec2)$ 
    by auto
          from  $\langle x \# M(\nu*(yvec1 @ y' \# yvec2)) \rangle \langle N \rangle$  have  $x \# M$  and  $x \# yvec1$  and
 $x \neq y'$  and  $x \# yvec2$  and  $x \# N$ 
          by simp+
          from  $\langle y \# M(\nu*(yvec1 @ y' \# yvec2)) \rangle \langle N \rangle$  have  $y \# M$  and  $y \# yvec1$  and
 $y \# yvec2$ 
          by simp+
          from  $\langle \Psi \triangleright ((y, y') \cdot (\nu x)P) \mapsto M(\nu*(yvec1 @ yvec2)) \rangle \langle N \rangle \prec P' \rangle \langle x \neq y \rangle \langle x \neq y' \rangle$ 
          have  $\Psi \triangleright (\nu x)((y, y') \cdot P) \mapsto M(\nu*(yvec1 @ yvec2)) \rangle \langle N \rangle \prec P'$  by(simp add: eqvts)

```

moreover note $\langle x \# \Psi \rangle$
moreover from $\langle x \# N \rangle \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \# M \rangle$ **have** $x \# M(\nu*(yvec1@yvec2))\langle N \rangle$ **by** *simp*
moreover note $\langle x \# P' \rangle$
moreover from $\langle yvec1 \#* \Psi \rangle \langle yvec2 \#* \Psi \rangle$ **have** $(yvec1@yvec2) \#* \Psi$ **by** *simp*
moreover from $\langle yvec1 \#* (\nu x)P \rangle \langle yvec2 \#* (\nu x)P \rangle \langle y \# yvec1 \rangle \langle y' \# yvec1 \rangle \langle y \# yvec2 \rangle \langle y' \# yvec2 \rangle \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle$
have $(yvec1@yvec2) \#* ((y, y') \cdot P)$ **by** *simp*
moreover from $\langle yvec1 \#* M \rangle \langle yvec2 \#* M \rangle$ **have** $(yvec1 @ yvec2) \#* M$ **by** *simp*
ultimately show *?case*
proof(*induct rule: resOutputCases''''*)
case(*cOpen M' xvec1 xvec2 x' N' P'*)
from $\langle M'(\nu*(xvec1 @ x' \# xvec2))\langle N' \rangle = M(\nu*(yvec1 @ yvec2))\langle N \rangle \rangle$
have $yvec1@yvec2 = xvec1@x'\#xvec2$ **and** $M = M'$ **and** $N = N'$ **by** (*simp add: action.inject*)
from $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle y' \# yvec1 \rangle \langle y' \# yvec2 \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle$
have $x \# (yvec1@yvec2)$ **and** $y \# (yvec1@yvec2)$ **and** $y' \# (yvec1@yvec2)$
by *simp+*
with $\langle yvec1@yvec2 = xvec1@x'\#xvec2 \rangle$
have $x \# xvec1$ **and** $x \neq x'$ **and** $x \# xvec2$ **and** $y \# xvec1$ **and** $y \neq x'$ **and** $y \# xvec2$
and $y' \# xvec1$ **and** $x' \neq y'$ **and** $y' \# xvec2$
by *auto*

show *?case*
proof(*cases x' \in set yvec1*)
assume $x' \in set yvec1$

with $\langle yvec1@yvec2 = xvec1@x'\#xvec2 \rangle \langle distinct (yvec1@yvec2) \rangle$
obtain $xvec2'$ **where** $Eq1: yvec1=xvec1@x'\#xvec2'$
and $Eq: xvec2=xvec2'@yvec2$
by(*metis partitionListLeft*)
from $\langle \Psi \triangleright ((x, x') \cdot [(y, y')] \cdot P) \mapsto M'(\nu*(xvec1@xvec2))\langle N' \rangle \prec P' \rangle$
 $\langle y' \in supp N \rangle \langle y' \# \Psi \rangle \langle y' \# M \rangle \langle y' \# xvec1 \rangle \langle y' \# xvec2 \rangle Eq \langle M=M' \rangle \langle N=N' \rangle$
have $\Psi \triangleright (\nu y')((x, x') \cdot [(y, y')] \cdot P) \mapsto M'(\nu*((xvec1@xvec2')@y'\#yvec2))\langle N' \rangle \prec P'$
by(*intro Open*) *auto*
then have $\Psi \triangleright (\nu x')((\nu y')((x, x') \cdot [(y, y')] \cdot P)) \mapsto M(\nu*(xvec1@x'\#xvec2'@y'\#yvec2))\langle N \rangle \prec P'$
using $\langle x' \in supp N' \rangle \langle x' \# \Psi \rangle \langle x' \# M' \rangle \langle x' \# xvec1 \rangle \langle x' \# xvec2 \rangle \langle x' \neq y' \rangle Eq \langle M=M' \rangle \langle N=N' \rangle$
by(*intro Open*) *auto*
with $\langle x' \neq y' \rangle \langle x \neq y' \rangle \langle x' \# [(y, y')] \cdot P \rangle$
have $\Psi \triangleright (\nu x)((\nu y')((x, x') \cdot [(y, y')] \cdot P)) \mapsto M(\nu*(xvec1@x'\#xvec2'@y'\#yvec2))\langle N \rangle \prec P'$
by(*subst alphaRes[where y=x']*) (*simp add: calc-atm eqts abs-fresh*)
with $Eq1 \langle y' \# (\nu x)P \rangle \langle x \neq y' \rangle R1$ **show** *?case*

```

    by(auto simp add: alphaRes abs-fresh)
  next
    assume  $\neg x' \in \text{set } yvec1$ 
    then have  $x' \# yvec1$  by(simp add: fresh-def)
    from  $\langle \neg x' \in \text{set } yvec1 \rangle \langle yvec1 @ yvec2 = xvec1 @ x' \# xvec2 \rangle$ 
    have  $x' \in \text{set } yvec2$ 
      by(auto simp add: append-eq-append-conv2 append-eq-Cons-conv)
    with  $\langle yvec1 @ yvec2 = xvec1 @ x' \# xvec2 \rangle \langle \text{distinct } (yvec1 @ yvec2) \rangle$ 
    obtain  $xvec2'$  where  $Eq: xvec1 = yvec1 @ xvec2'$ 
      and  $Eq1: yvec2 = xvec2' @ x' \# xvec2$ 
    by(metis partitionListRight)
    from  $\langle \Psi \triangleright ([x, x'] \cdot [(y, y') \cdot P]) \mapsto M'(\nu^*(xvec1 @ xvec2)) \langle N' \rangle \prec P' \rangle$ 
     $\langle y' \in \text{supp } N \rangle \langle y' \# \Psi \rangle \langle y' \# M \rangle \langle y' \# xvec1 \rangle \langle y' \# xvec2 \rangle Eq \langle M = M' \rangle \langle N = N' \rangle$ 
    have  $\Psi \triangleright (\nu y') \langle [(x, x') \cdot [(y, y') \cdot P]) \mapsto M'(\nu^*(yvec1 @ y' \# xvec2' @ xvec2)) \langle N' \rangle$ 
     $\prec P'$ 
      by(intro Open) (assumption | simp)+
    then have  $\Psi \triangleright (\nu x') \langle (\nu y') \langle [(x, x') \cdot [(y, y') \cdot P]) \mapsto M(\nu^*((yvec1 @ y' \# xvec2') @ x' \# xvec2)) \langle N \rangle$ 
     $\prec P'$ 
      using  $\langle x' \in \text{supp } N' \rangle \langle x' \# \Psi \rangle \langle x' \# M' \rangle \langle x' \# xvec1 \rangle \langle x' \# xvec2 \rangle \langle x' \neq y' \rangle$ 
       $Eq \langle M = M' \rangle \langle N = N' \rangle$ 
      by(intro Open) auto
    with  $\langle x' \neq y' \rangle \langle x \neq y' \rangle \langle x' \# [(y, y') \cdot P] \rangle$ 
    have  $\Psi \triangleright (\nu x) \langle (\nu y') \langle [(y, y') \cdot P]) \mapsto M(\nu^*((yvec1 @ y' \# xvec2') @ x' \# xvec2)) \langle N \rangle$ 
     $\prec P'$ 
      by(subst alphaRes[where  $y = x'$ ]) (simp add: calc-atm eqts abs-fresh)+
    with  $Eq1 \langle y' \# (\nu x) P \rangle \langle x \neq y' \rangle R1$  show ?case
    by(auto simp add: alphaRes abs-fresh)
  qed
next
case(cRes P')
  from  $\langle \Psi \triangleright ([y, y'] \cdot P) \mapsto M(\nu^*(yvec1 @ yvec2)) \langle N \rangle \prec P' \rangle \langle y' \in \text{supp } N \rangle$ 
   $\langle y' \# \Psi \rangle \langle y' \# M \rangle \langle y' \# yvec1 \rangle \langle y' \# yvec2 \rangle$ 
  have  $\Psi \triangleright (\nu y') \langle [(y, y') \cdot P] \mapsto M(\nu^*(yvec1 @ y' \# yvec2)) \langle N \rangle \prec P' \rangle$  by(rule Open)
  with  $\langle y' \# (\nu x) P \rangle \langle x \neq y' \rangle$  have  $\Psi \triangleright (\nu y) P \mapsto M(\nu^*(yvec1 @ y' \# yvec2)) \langle N \rangle$ 
   $\prec P'$  by(simp add: alphaRes abs-fresh)
  then have  $\Psi \triangleright (\nu x) \langle (\nu y) P \mapsto M(\nu^*(yvec1 @ y' \# yvec2)) \langle N \rangle \prec (\nu x) P'$ 
  using  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# yvec1 \rangle \langle x \neq y' \rangle \langle x \# yvec2 \rangle \langle x \# N \rangle$ 
  by(intro Scope) auto
  moreover have  $(\Psi, (\nu x) P', (\nu x) P') \in \text{Rel}$  by(rule R1)
  ultimately show ?case by blast
  qed
next
case(cBrOpen M yvec1 yvec2 y' N P')
  from  $\langle yvec1 \#* yvec2 \rangle \langle \text{distinct } yvec1 \rangle \langle \text{distinct } yvec2 \rangle$  have  $\text{distinct}(yvec1 @ yvec2)$ 
  by auto
  from  $\langle x \# M(\nu^*(yvec1 @ y' \# yvec2)) \langle N \rangle \rangle$  have  $x \# M$  and  $x \# yvec1$  and
   $x \neq y'$  and  $x \# yvec2$  and  $x \# N$ 
  by simp+

```

from $\langle y \# \text{;M}(\nu^*(yvec1 @ y' \# yvec2)) \rangle \langle N \rangle$ **have** $y \# M$ **and** $y \# yvec1$ **and**
 $y \# yvec2$
by *simp+*
from $\langle \Psi \triangleright ((y, y') \cdot (\nu x)P) \mapsto \text{;M}(\nu^*(yvec1 @ yvec2)) \rangle \langle N \rangle \prec P'$ $\langle x \neq y \rangle$
 $\langle x \neq y' \rangle$
have $\Psi \triangleright (\nu x)((y, y') \cdot P) \mapsto \text{;M}(\nu^*(yvec1 @ yvec2)) \rangle \langle N \rangle \prec P'$ **by** (*simp*
add: eqvts)
moreover note $\langle x \# \Psi \rangle$
moreover from $\langle x \# N \rangle \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle x \# M \rangle$ **have** $x \#$
 $\text{;M}(\nu^*(yvec1 @ yvec2)) \rangle \langle N \rangle$ **by** *simp*
moreover note $\langle x \# P' \rangle$
moreover from $\langle yvec1 \# \Psi \rangle \langle yvec2 \# \Psi \rangle$ **have** $(yvec1 @ yvec2) \# \Psi$ **by**
simp
moreover from $\langle yvec1 \# (\nu x)P \rangle \langle yvec2 \# (\nu x)P \rangle \langle y \# yvec1 \rangle \langle y' \# yvec1 \rangle$
 $\langle y \# yvec2 \rangle \langle y' \# yvec2 \rangle \langle x \# yvec1 \rangle \langle x \# yvec2 \rangle$
have $(yvec1 @ yvec2) \# ((y, y') \cdot P)$ **by** *simp*
moreover from $\langle yvec1 \# M \rangle \langle yvec2 \# M \rangle$ **have** $(yvec1 @ yvec2) \# M$
by *simp*
ultimately show *?case*
proof (*induct rule: resBrOutputCases'*)
case (*cBrOpen M' xvec1 xvec2 x' N' P'*)
from $\text{;M}'(\nu^*(xvec1 @ x' \# xvec2)) \rangle \langle N' \rangle = \text{;M}(\nu^*(yvec1 @ yvec2)) \rangle \langle N \rangle$
have $yvec1 @ yvec2 = xvec1 @ x' \# xvec2$ **and** $M = M'$ **and** $N = N'$ **by** (*simp add:*
action.inject)
from $\langle x \# yvec1 \rangle \langle x \# yvec2 \rangle \langle y' \# yvec1 \rangle \langle y' \# yvec2 \rangle \langle y \# yvec1 \rangle \langle y \# yvec2 \rangle$
have $x \# (yvec1 @ yvec2)$ **and** $y \# (yvec1 @ yvec2)$ **and** $y' \# (yvec1 @ yvec2)$
by *simp+*
with $\langle yvec1 @ yvec2 = xvec1 @ x' \# xvec2 \rangle$
have $x \# xvec1$ **and** $x \neq x'$ **and** $x \# xvec2$ **and** $y \# xvec1$ **and** $y \neq x'$ **and**
 $y \# xvec2$
and $y' \# xvec1$ **and** $x' \neq y'$ **and** $y' \# xvec2$
by *auto*

show *?case*
proof (*cases x' \in set yvec1*)
assume $x' \in \text{set } yvec1$

with $\langle yvec1 @ yvec2 = xvec1 @ x' \# xvec2 \rangle \langle \text{distinct } (yvec1 @ yvec2) \rangle$
obtain $xvec2'$ **where** $Eq1: yvec1 = xvec1 @ x' \# xvec2'$
and $Eq: xvec2 = xvec2' @ yvec2$
by (*metis partitionListLeft*)
from $\langle \Psi \triangleright ((x, x') \cdot [(y, y')] \cdot P) \mapsto \text{;M}'(\nu^*(xvec1 @ xvec2)) \rangle \langle N' \rangle \prec P'$
 $\langle y' \in \text{supp } N \rangle \langle y' \# \Psi \rangle \langle y' \# M \rangle \langle y' \# xvec1 \rangle \langle y' \# xvec2 \rangle Eq \langle M = M' \rangle \langle N = N' \rangle$
have $\Psi \triangleright (\nu y')((x, x') \cdot [(y, y')] \cdot P) \mapsto \text{;M}'(\nu^*((xvec1 @ xvec2') @ y' \# yvec2)) \rangle \langle N' \rangle$
 $\prec P'$
by (*intro BrOpen*) *auto*
then have $\Psi \triangleright (\nu x')((\nu y')((x, x') \cdot [(y, y')] \cdot P)) \mapsto \text{;M}(\nu^*(xvec1 @ x' \# xvec2' @ y' \# yvec2)) \rangle \langle N \rangle$
 $\prec P'$
using $\langle x' \in \text{supp } N' \rangle \langle x' \# \Psi \rangle \langle x' \# M' \rangle \langle x' \# xvec1 \rangle \langle x' \# xvec2 \rangle \langle x' \neq$

```

y'⟩ Eq ⟨M=M'⟩ ⟨N=N'⟩
  by(intro BrOpen) auto
  with ⟨x' ≠ y'⟩ ⟨x ≠ y'⟩ ⟨x' # [(y, y')] · P⟩
  have Ψ ▷ (νx)((νy')([(y, y')] · P)) ⟶iM(ν*(xvec1@x'#xvec2'@y'#yvec2))⟨N⟩
  < P'
    by(subst alphaRes[where y=x'] (simp add: calc-atm eqts abs-fresh)+
    with Eq1 ⟨y' # (νx)P⟩ ⟨x ≠ y'⟩ R1 show ?case
    by(auto simp add: alphaRes abs-fresh)
  next
  assume ¬x' ∈ set yvec1
  then have x' # yvec1 by(simp add: fresh-def)
  from ⟨¬x' ∈ set yvec1⟩ ⟨yvec1@yvec2 = xvec1@x'#xvec2⟩
  have x' ∈ set yvec2
    by(auto simp add: append-eq-append-conv2 append-eq-Cons-conv)
  with ⟨yvec1@yvec2 = xvec1@x'#xvec2⟩ ⟨distinct (yvec1@yvec2)⟩
  obtain xvec2' where Eq: xvec1=yvec1@xvec2'
    and Eq1: yvec2=xvec2'@x'#xvec2
    by(metis partitionListRight)
  from ⟨Ψ ▷ [(x, x')] · [(y, y')] · P ⟶iM'(ν*(xvec1@xvec2))⟨N'⟩ < P'⟩
  ⟨y' ∈ supp N⟩ ⟨y' # Ψ⟩ ⟨y' # M⟩ ⟨y' # xvec1⟩ ⟨y' # xvec2⟩ Eq ⟨M=M'⟩ ⟨N = N'⟩
  have Ψ ▷ (νy')([(x, x')] · [(y, y')] · P) ⟶iM'(ν*(yvec1@y'#xvec2'@xvec2))⟨N'⟩
  < P'
    by(intro BrOpen) (assumption | simp)+
  then have Ψ ▷ (νx')((νy')([(x, x')] · [(y, y')] · P)) ⟶iM(ν*((yvec1@y'#xvec2')@x'#xvec2))⟨N⟩
  < P'
    using ⟨x' ∈ supp N'⟩ ⟨x' # Ψ⟩ ⟨x' # M'⟩ ⟨x' # xvec1⟩ ⟨x' # xvec2⟩ ⟨x' ≠
y'⟩ Eq ⟨M=M'⟩ ⟨N=N'⟩
  by(intro BrOpen) auto
  with ⟨x' ≠ y'⟩ ⟨x ≠ y'⟩ ⟨x' # [(y, y')] · P⟩
  have Ψ ▷ (νx)((νy')([(y, y')] · P)) ⟶iM(ν*((yvec1@y'#xvec2')@x'#xvec2))⟨N⟩
  < P'
    by(subst alphaRes[where y=x'] (simp add: calc-atm eqts abs-fresh)+
    with Eq1 ⟨y' # (νx)P⟩ ⟨x ≠ y'⟩ R1 show ?case
    by(auto simp add: alphaRes abs-fresh)
  qed
next
case(cRes P')
  from ⟨Ψ ▷ [(y, y')] · P ⟶iM(ν*(yvec1@yvec2))⟨N⟩ < P'⟩ ⟨y' ∈ supp
N⟩ ⟨y' # Ψ⟩ ⟨y' # M⟩ ⟨y' # yvec1⟩ ⟨y' # yvec2⟩
  have Ψ ▷ (νy')([(y, y')] · P) ⟶iM(ν*(yvec1@y'#yvec2))⟨N⟩ < P' by(rule
BrOpen)
  with ⟨y' # (νx)P⟩ ⟨x ≠ y'⟩ have Ψ ▷ (νy)P ⟶iM(ν*(yvec1@y'#yvec2))⟨N⟩
  < P' by(simp add: alphaRes abs-fresh)
  then have Ψ ▷ (νx)((νy)P) ⟶iM(ν*(yvec1@y'#yvec2))⟨N⟩ < (νx)P'
  using ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # yvec1⟩ ⟨x ≠ y'⟩ ⟨x # yvec2⟩ ⟨x # N⟩
  by(intro Scope) auto
  moreover have (Ψ, (νx)P', (νx)P') ∈ Rel by(rule R1)
  ultimately show ?case by blast
  qed

```

next
case(*cRes* P')
from $\langle x \# (\nu y)P' \rangle \langle x \neq y \rangle$ **have** $x \# P'$ **by**(*simp add: abs-fresh*)
with $\langle \Psi \triangleright (\nu x)P \mapsto \alpha \prec P' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle$
show $?case$ **using** $\langle bn \ \alpha \ \#* \ \Psi \rangle \langle bn \ \alpha \ \#* \ P \rangle \langle bn \ \alpha \ \#* \ subject \ \alpha \rangle \langle y \ \# \ \alpha \rangle$
proof(*induct rule: resCases'*[**where** $C=(x, y)$])
case(*cOpen* $M \ xvec1 \ xvec2 \ x' \ N \ P'$)
from $\langle y \ \# \ M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$ **have** $y \neq x'$ **and** $y \ \# \ M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle$
by *simp+*
from $\langle \Psi \triangleright ([x, x'] \cdot P) \mapsto M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle \prec P' \rangle \langle y \ \# \ \Psi \rangle \langle y \ \# \ M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle$
have $\Psi \triangleright (\nu y)([x, x'] \cdot P) \mapsto M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle \prec (\nu y)P'$
by(*rule Scope*)
then have $\Psi \triangleright (\nu x')((\nu y)([x, x'] \cdot P)) \mapsto M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$
 $\prec (\nu y)P'$
using $\langle x' \in supp \ N \rangle \langle x' \ \# \ \Psi \rangle \langle x' \ \# \ M \rangle \langle x' \ \# \ xvec1 \rangle \langle x' \ \# \ xvec2 \rangle$
by(*rule Open*)
with $\langle y \neq x' \rangle \langle x \neq y \rangle \langle x' \ \# \ P \rangle$ **have** $\Psi \triangleright (\nu x)((\nu y)P) \mapsto M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$
 $\prec (\nu y)P'$
by(*subst alphaRes*[**where** $y=x'$]) (*simp add: abs-fresh eqts calc-atm*)+
moreover have $(\Psi, (\nu y)P', (\nu y)P') \in Rel$ **by**(*rule R1*)
ultimately show $?case$ **by** *blast*
next
case(*cBrOpen* $M \ xvec1 \ xvec2 \ x' \ N \ P'$)
from $\langle y \ \# \ M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$ **have** $y \neq x'$ **and** $y \ \# \ M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle$
by *simp+*
from $\langle \Psi \triangleright ([x, x'] \cdot P) \mapsto M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle \prec P' \rangle \langle y \ \# \ \Psi \rangle \langle y \ \# \ M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle$
 $\prec M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle$
have $\Psi \triangleright (\nu y)([x, x'] \cdot P) \mapsto M(\nu*(xvec1 @ xvec2)) \rangle \langle N \rangle \prec (\nu y)P'$
by(*rule Scope*)
then have $\Psi \triangleright (\nu x')((\nu y)([x, x'] \cdot P)) \mapsto M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$
 $\prec (\nu y)P'$
using $\langle x' \in supp \ N \rangle \langle x' \ \# \ \Psi \rangle \langle x' \ \# \ M \rangle \langle x' \ \# \ xvec1 \rangle \langle x' \ \# \ xvec2 \rangle$
by(*rule BrOpen*)
with $\langle y \neq x' \rangle \langle x \neq y \rangle \langle x' \ \# \ P \rangle$ **have** $\Psi \triangleright (\nu x)((\nu y)P) \mapsto M(\nu*(xvec1 @ x' \# xvec2)) \rangle \langle N \rangle$
 $\prec (\nu y)P'$
by(*subst alphaRes*[**where** $y=x'$]) (*simp add: abs-fresh eqts calc-atm*)+
moreover have $(\Psi, (\nu y)P', (\nu y)P') \in Rel$ **by**(*rule R1*)
ultimately show $?case$ **by** *blast*
next
case(*cRes* P')
from $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle y \ \# \ \Psi \rangle \langle y \ \# \ \alpha \rangle$
have $\Psi \triangleright (\nu y)P \mapsto \alpha \prec (\nu y)P'$ **by**(*rule Scope*)
then have $\Psi \triangleright (\nu x)((\nu y)P) \mapsto \alpha \prec (\nu x)((\nu y)P')$ **using** $\langle x \ \# \ \Psi \rangle \langle x \ \# \ \alpha \rangle$
by(*rule Scope*)
moreover from $\langle x \ \# \ \Psi \rangle \langle y \ \# \ \Psi \rangle$ **have** $(\Psi, (\nu x)((\nu y)P'), (\nu y)((\nu x)P')) \in Rel$
Rel
by(*rule R2*)
ultimately show $?case$ **by** *blast*

```

next
  case(cBrClose M xvec N P')
  then show ?case
  proof(cases  $y \# \downarrow M(\downarrow \nu^*xvec)\langle N \rangle$ )
    case True
    with  $\langle \Psi \triangleright P \mapsto \downarrow M(\downarrow \nu^*xvec)\langle N \rangle \prec P' \rangle$ 
    have  $\Psi \triangleright (\downarrow \nu y)P \mapsto \downarrow M(\downarrow \nu^*xvec)\langle N \rangle \prec (\downarrow \nu y)P'$  using  $\langle y \# \Psi \rangle$ 
      by(intro Scope)
    then have  $\Psi \triangleright (\downarrow \nu x)((\downarrow \nu y)P) \mapsto \tau \prec ((\downarrow \nu x)((\downarrow \nu^*xvec)(\downarrow \nu y)P'))$  using  $\langle x$ 
   $\in \text{supp } M \rangle \langle x \# \Psi \rangle$ 
      by(rule BrClose)
    moreover have  $(\Psi, ((\downarrow \nu x)((\downarrow \nu^*xvec)(\downarrow \nu y)P')), (\downarrow \nu y)((\downarrow \nu x)((\downarrow \nu^*xvec)P')) \in$ 
  Rel
    proof –
      have  $\text{mset } (x\#xvec@[y]) = \text{mset } (y\#x\#xvec)$ 
        by simp
      moreover have  $(x\#xvec@[y]) \#* \Psi$  using  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle xvec \#* \Psi \rangle$ 
        by simp
      ultimately have  $(\Psi, (\downarrow \nu^*(x\#xvec@[y]))P', (\downarrow \nu^*(y\#x\#xvec))P') \in \text{Rel}$ 
        by(metis R3)
      then show ?thesis
        by(auto simp add: resChain.Append)
    qed
    ultimately show ?thesis
      by blast
  next
  case False
  then have  $y \in \text{supp}(\downarrow M(\downarrow \nu^*xvec)\langle N \rangle)$  unfolding fresh-def by simp
  show ?thesis
  proof(cases  $y \in \text{supp}(M)$ )
    case True
    with  $\langle \Psi \triangleright P \mapsto \downarrow M(\downarrow \nu^*xvec)\langle N \rangle \prec P' \rangle$ 
    have  $\Psi \triangleright (\downarrow \nu y)P \mapsto \tau \prec (\downarrow \nu y)((\downarrow \nu^*xvec)P')$  using  $\langle y \# \Psi \rangle$ 
      by(rule BrClose)
    then have  $\Psi \triangleright (\downarrow \nu x)((\downarrow \nu y)P) \mapsto \tau \prec (\downarrow \nu x)((\downarrow \nu y)((\downarrow \nu^*xvec)P'))$  using  $\langle x$ 
   $\# \Psi \rangle$ 
      by(rule Scope simp)
    moreover have  $(\Psi, (\downarrow \nu x)((\downarrow \nu y)((\downarrow \nu^*xvec)P')), (\downarrow \nu y)((\downarrow \nu x)((\downarrow \nu^*xvec)P')) \in \text{Rel}$ 
   $\in \text{Rel}$  using  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ 
      by(metis R2)
    ultimately show ?thesis
      by blast
  next
  case False
  then have  $y \# M$  by(simp add: fresh-def)
  from  $\langle xvec \#* (x, y) \rangle$  have  $y \# xvec$  by simp
  with False  $\langle y \in \text{supp}(\downarrow M(\downarrow \nu^*xvec)\langle N \rangle) \rangle$ 
  have  $y \in \text{supp } N$ 
    by(simp add: fresh-def action.supp)

```

```

    from  $\langle \Psi \triangleright P \mapsto_{iM}(\nu^*xvec)\langle N \rangle \prec P' \rangle$  have  $\Psi \triangleright P \mapsto_{iM}(\nu^*([]@xvec))\langle N \rangle$ 
 $\prec P'$ 
      by simp
      then have  $\Psi \triangleright (\nu y)P \mapsto_{iM}(\nu^*([]@y\#xvec))\langle N \rangle \prec P'$  using  $\langle y \in$ 
 $supp\ N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec \rangle$ 
      by(intro BrOpen) (assumption|simp)+
      then have  $\Psi \triangleright (\nu y)P \mapsto_{iM}(\nu^*(y\#xvec))\langle N \rangle \prec P'$ 
      by simp
      then have  $\Psi \triangleright (\nu x)((\nu y)P) \mapsto_{\tau} \prec (\nu x)((\nu y)((\nu^*xvec)P'))$  using  $\langle x$ 
 $\in supp\ M \rangle \langle x \# \Psi \rangle$ 
      by(auto dest: BrClose)
      moreover have  $(\Psi, (\nu x)((\nu y)((\nu^*xvec)P')), (\nu y)((\nu x)((\nu^*xvec)P'))$ 
 $\in Rel$  using  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$ 
      by(rule R2)
      ultimately show ?thesis by blast
    qed
  qed
  qed
next
  case(cBrClose M xvec N P')
  from  $\langle xvec \#* x \rangle$  have  $x \# xvec$  by simp
  have  $x \# (\nu x)P$  by(simp add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst,
OF fin-supp])
  have  $x \# P'$ 
  by(rule brotputFreshDerivativeP) fact+
  have  $x \# N$ 
  by(rule brotputFreshDerivativeN) fact+
  moreover from  $\langle \Psi \triangleright (\nu x)P \mapsto_{iM}(\nu^*xvec)\langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \#$ 
 $(\nu x)P \rangle$  have  $x \# M$ 
  by(rule brOutputFreshSubject)
  moreover note  $\langle x \# xvec \rangle$ 
  ultimately have  $x \# iM(\nu^*xvec)\langle N \rangle$ 
  by simp
  have  $bn\ (iM(\nu^*xvec)\langle N \rangle) \#* \Psi$  using  $\langle xvec \#* \Psi \rangle$  by simp
  have  $bn\ (iM(\nu^*xvec)\langle N \rangle) \#* P$  using  $\langle xvec \#* (\nu x)P \rangle \langle x \# xvec \rangle$  by(simp
add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst, OF fin-supp])
  have  $bn\ (iM(\nu^*xvec)\langle N \rangle) \#* subject\ (iM(\nu^*xvec)\langle N \rangle)$  using  $\langle xvec \#* M \rangle$  by
simp
  have  $y \in supp(subject\ (iM(\nu^*xvec)\langle N \rangle))$  using  $\langle y \in supp\ M \rangle$ 
  by(simp add: supp-some)
  obtain  $M' xvec' N'$  where  $iM(\nu^*xvec)\langle N \rangle = iM'(\nu^*xvec')\langle N' \rangle$ 
  by auto
  from  $\langle \Psi \triangleright (\nu x)P \mapsto_{iM}(\nu^*xvec)\langle N \rangle \prec P' \rangle \langle x \# \Psi \rangle \langle x \# iM(\nu^*xvec)\langle N \rangle \rangle \langle x$ 
 $\# P' \rangle \langle bn\ (iM(\nu^*xvec)\langle N \rangle) \#* \Psi \rangle \langle bn\ (iM(\nu^*xvec)\langle N \rangle) \#* P \rangle \langle bn\ (iM(\nu^*xvec)\langle N \rangle)$ 
 $\#* subject\ (iM(\nu^*xvec)\langle N \rangle) \rangle \langle iM(\nu^*xvec)\langle N \rangle = iM'(\nu^*xvec')\langle N' \rangle \rangle \langle y \in supp(subject$ 
 $(iM(\nu^*xvec)\langle N \rangle)) \rangle$ 
  have  $\exists Q'. \Psi \triangleright (\nu x)((\nu y)P) \mapsto_{\tau} \prec Q' \wedge (\Psi, Q', (\nu y)((\nu^*(bn\ (iM(\nu^*xvec)\langle N \rangle)))P'))$ 
 $\in Rel$ 
  proof(induct rule: resCases'[where C=y])

```



```

    case cOpen then show ?case by(simp add: residualInject)
  next
    case(cBrOpen M xvec yvec z N P')
    from ⟨y ∈ supp (subject (iM(ν*(xvec @ z # yvec))⟨N⟩))⟩ have y ∈ supp M
      by(simp add: supp-some)
    then have y ≠ z using ⟨z # M⟩ by(auto simp add: fresh-def)
    from ⟨Ψ ▷ [(x, z)] · P ⟶ iM(ν*(xvec @ yvec))⟨N⟩ < P'⟩ ⟨y ∈ supp M⟩
    ⟨y # Ψ⟩
    have Ψ ▷ (νy)((x, z) · P) ⟶ τ < (νy)((ν*(xvec@yvec))P')
      by(rule BrClose)
    then have Ψ ▷ (νz)((νy)((x, z) · P)) ⟶ τ < (νz)((νy)((ν*(xvec@yvec))P'))
  using ⟨z # Ψ⟩
    by(rule Scope) simp
    then have Ψ ▷ (νx)((νy)P) ⟶ τ < (νz)((νy)((ν*(xvec@yvec))P'))
  using ⟨z # P⟩ ⟨x ≠ y⟩ ⟨y ≠ z⟩
    apply(subst alphaRes[where x=x and y=z])
    apply(simp add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst, OF
  fin-supp])
    apply(simp add: eqts swap-simps)
    done
    moreover have (Ψ, (νz)((νy)((ν*(xvec @ yvec))P')), (νy)((ν*bn (iM(ν*(xvec
  @ z # yvec))⟨N⟩))P')) ∈ Rel
    proof -
      have mset(z#y#xvec@yvec) = mset(y#xvec@z#yvec)
        by simp
      moreover have (z#y#xvec@yvec) #* Ψ using ⟨z # Ψ⟩ ⟨y # Ψ⟩ ⟨xvec #*
  Ψ⟩ ⟨yvec #* Ψ⟩
        by simp
      ultimately have (Ψ, (ν*(z#y#xvec@yvec))P', (ν*(y#xvec@z#yvec))P')
  ∈ Rel
        by(metis R3)
      then show ?thesis
        by(simp add: resChain.Append)
    qed
    ultimately show ?case
      by blast
  next
    case(cRes P')
    from ⟨Ψ ▷ P ⟶ iM(ν*xvec)⟨N⟩ < P'⟩ ⟨y ∈ supp M⟩ ⟨y # Ψ⟩
    have Ψ ▷ (νy)P ⟶ τ < (νy)((ν*xvec)P')
      by(rule BrClose)
    then have Ψ ▷ (νx)((νy)P) ⟶ τ < (νx)((νy)((ν*xvec)P')) using ⟨x #
  Ψ⟩
      by(rule Scope) simp
    moreover have (Ψ, (νx)((νy)((ν*xvec)P')), (νy)((ν*bn (iM(ν*xvec)⟨N⟩))
  (νx)P'))
  ∈ Rel
    proof -
      have mset(x#y#xvec) = mset(y#xvec@[x])
        by simp

```

```

moreover have  $(x\#y\#xvec) \#* \Psi$  using  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle xvec \#* \Psi \rangle$ 
by simp
ultimately have  $(\Psi, (\nu*(x\#y\#xvec))P', (\nu*(y\#xvec@[x]))P') \in Rel$ 
by(metis R3)
then show ?thesis by(simp add: resChainAppend)
qed
ultimately show ?case
by blast
next
case cBrClose then show ?case by simp
qed
then show ?case by simp
qed
qed
qed

```

lemma *parAssocLeft*:

```

fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) psi$ 
and  $Q :: ('a, 'b, 'c) psi$ 
and  $R :: ('a, 'b, 'c) psi$ 
and  $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

```

assumes *eqvt Rel*

```

and  $C1: \bigwedge \Psi' S T U. (\Psi, (S \parallel T) \parallel U, S \parallel (T \parallel U)) \in Rel$ 
and  $C2: \bigwedge xvec \Psi' S T U. \llbracket xvec \#* \Psi'; xvec \#* S \rrbracket \implies (\Psi', (\nu*xvec)((S \parallel T) \parallel U), S \parallel ((\nu*xvec)(T \parallel U))) \in Rel$ 
and  $C3: \bigwedge xvec \Psi' S T U. \llbracket xvec \#* \Psi'; xvec \#* U \rrbracket \implies (\Psi', ((\nu*xvec)(S \parallel T)) \parallel U, (\nu*xvec)(S \parallel (T \parallel U))) \in Rel$ 
and  $C4: \bigwedge \Psi' S T xvec. \llbracket (\Psi', S, T) \in Rel; xvec \#* \Psi' \rrbracket \implies (\Psi', (\nu*xvec)S, (\nu*xvec)T) \in Rel$ 

```

shows $\Psi \triangleright (P \parallel Q) \parallel R \rightsquigarrow_{[Rel]} P \parallel (Q \parallel R)$

using *<eqvt Rel>*

proof(*induct rule: simI[of - - - ()]*)

case(*cSim α PQR*)

from $\langle bn \alpha \#* (P \parallel Q \parallel R) \rangle$ **have** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ **and** $bn \alpha \#* R$ **by** *simp+*

then have $bn \alpha \#* (Q \parallel R)$ **by** *simp*

with $\langle \Psi \triangleright P \parallel (Q \parallel R) \mapsto_{\alpha} \prec PQR \rangle$ $\langle bn \alpha \#* \Psi \rangle$ $\langle bn \alpha \#* P \rangle$

show *?case* **using** *<bn α $\#*$ subject α >*

proof(*induct rule: parCases[where $C = (\Psi, P, Q, R, \alpha)$]*)

case(*cPar1 $P' A_{QR} \Psi_{QR}$*)

from $\langle A_{QR} \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \#* Q$ **and** $A_{QR} \#* R$ **by** *simp+*

with $\langle extractFrame(Q \parallel R) = \langle A_{QR}, \Psi_{QR} \rangle \langle distinct A_{QR} \rangle$

obtain $A_Q \Psi_Q A_R \Psi_R$ **where** $A_{QR} = A_Q @ A_R$ **and** $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **and**

FrQ: $extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** *FrR*: $extractFrame R = \langle A_R, \Psi_R \rangle$

and $A_Q \#* \Psi_R$ **and** $A_R \#* \Psi_Q$

by(*auto intro: mergeFrameE dest: extractFrameFreshChain*)

from $\langle A_{QR} = A_Q @ A_R \rangle \langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* Q \rangle \langle A_{QR} \#* \alpha \rangle$
have $A_Q \#* \Psi$ **and** $A_R \#* \Psi$ **and** $A_Q \#* P$ **and** $A_R \#* P$ **and** $A_Q \#* Q$ **and**
 $A_R \#* Q$ **and** $A_Q \#* \alpha$ **and** $A_R \#* \alpha$
by *simp+*

from $\langle \Psi \otimes \Psi_{QR} \triangleright P \mapsto \alpha \prec P' \rangle \langle \Psi_{QR} = \Psi_Q \otimes \Psi_R \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q$
 $\triangleright P \mapsto \alpha \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \alpha \prec (P' \parallel Q)$ **using** *FrQ* $\langle bn \alpha \#* Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* \alpha \rangle$
by(*intro Par1*) *auto*
then have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \alpha \prec ((P' \parallel Q) \parallel R)$ **using** *FrR* $\langle bn \alpha \#* R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* \alpha \rangle$
by(*auto intro: Par1*)
moreover have $(\Psi, (P' \parallel Q) \parallel R, P' \parallel (Q \parallel R)) \in Rel$ **by**(*rule C1*)
ultimately show *?case by blast*

next

case(*cPar2 QR AP PsiP*)
from $\langle A_P \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_P \#* Q$ **and** $A_P \#* R$ **and** $A_P \#* \alpha$
by *simp+*
have *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by** *fact*
with $\langle bn \alpha \#* P \rangle \langle A_P \#* \alpha \rangle$ **have** $bn \alpha \#* \Psi_P$ **by**(*auto dest: extractFrame-FreshChain*)
with $\langle bn \alpha \#* \Psi \rangle$ **have** $bn \alpha \#* (\Psi \otimes \Psi_P)$ **by** *force*
with $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto \alpha \prec QR \rangle$
show *?case using* $\langle bn \alpha \#* Q \rangle \langle bn \alpha \#* R \rangle \langle bn \alpha \#* subject \alpha \rangle \langle A_P \#* Q \rangle \langle A_P \#* R \rangle$
proof(*induct rule: parCasesSubject[where C = (AP, PsiP, P, Q, R, Psi)]*)
case(*cPar1 Q' AR PsiR*)
from $\langle A_R \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \#* A_P$ **and** $A_R \#* P$ **and** $A_R \#* Q$
and $A_R \#* \Psi_P$ **and** $A_R \#* \Psi$
by *simp+*
from $\langle A_P \#* R \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* A_P \rangle$ **have** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)
from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto \alpha \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto \alpha$
 $\prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$
using *FrP* $\langle bn \alpha \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_P \#* Q \rangle \langle A_P \#* \alpha \rangle$
by(*intro Par2*) (*assumption | force*)
then have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \alpha \prec ((P \parallel Q') \parallel R)$
using $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle bn \alpha \#* R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle$
 $\langle A_R \#* \alpha \rangle$
by(*intro Par1*) (*assumption | simp*)
moreover have $(\Psi, (P \parallel Q') \parallel R, P \parallel (Q' \parallel R)) \in Rel$ **by**(*rule C1*)
ultimately show *?case by blast*

next

case(*cPar2* $R' A_Q \Psi_Q$)
from $\langle A_Q \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_Q \#* A_P$ **and** $A_Q \#* R$ **and** $A_Q \#* \Psi_P$ **and** $A_Q \#* \Psi$
by *simp+*
have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
from $\langle A_P \#* Q \rangle$ *FrQ* $\langle A_Q \#* A_P \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto \alpha \prec R' \rangle$
have $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto \alpha \prec R'$
by(*blast intro: statEqTransition Associativity*)
moreover from *FrP FrQ* $\langle A_Q \#* A_P \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have *extractFrame*($P \parallel Q$) = $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp*
moreover from $\langle bn \ \alpha \ \#* P \rangle \langle bn \ \alpha \ \#* Q \rangle$ **have** $bn \ \alpha \ \#* (P \parallel Q)$ **by** *simp*
moreover from $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $(A_P @ A_Q) \#* \Psi$ **by** *simp*
moreover from $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle$ **have** $(A_P @ A_Q) \#* R$ **by** *simp*
moreover from $\langle A_P \#* \alpha \rangle \langle A_Q \#* \alpha \rangle$ **have** $(A_P @ A_Q) \#* \alpha$ **by** *simp*
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \alpha \prec ((P \parallel Q) \parallel R')$
by(*rule Par2*)
moreover have $(\Psi, (P \parallel Q) \parallel R', P \parallel (Q \parallel R')) \in Rel$ **by**(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cComm1* $\Psi_R M N Q' A_Q \Psi_Q K \text{vec } R' A_R$)
from $\langle A_Q \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$
have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* R$ **and** $A_Q \#* A_P$ **and** $A_Q \#* \Psi_P$
and $A_Q \#* \Psi$ **by** *simp+*
from $\langle A_R \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* Q$ **and** $A_R \#* R$ **and** $A_R \#* A_P$ **and** $A_R \#* \Psi$ **by** *simp+*
from $\langle \text{vec} \ \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $\text{vec} \ \#* A_P$ **and** $\text{vec} \ \#* P$ **and** $\text{vec} \ \#* Q$ **and** $\text{vec} \ \#* \Psi$ **by** *simp+*

have *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
have *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with $\langle A_P \#* R \rangle \langle A_R \#* A_P \rangle$ **have** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)
from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto K(\nu * \text{vec}) \langle N \rangle \prec R' \rangle \langle A_P \#* R \rangle \langle \text{vec} \ \#* A_P \rangle \langle \text{vec} \ \#* K \rangle \langle \text{distinct vec} \rangle$ **have** $A_P \#* N$
by(*auto intro: outputFreshChainDerivative*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto M \langle N \rangle \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto M \langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto M \langle N \rangle \prec (P \parallel Q')$ **using** *FrP* $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$
by(*intro Par2*) *auto*
moreover from *FrP FrQ* $\langle A_P \#* \Psi_Q \rangle \langle A_Q \#* A_P \rangle \langle A_Q \#* \Psi_P \rangle$ **have** *extractFrame*($P \parallel Q$) = $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp*

moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto K(\nu^*xvec)\langle N \rangle \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \triangleright R \mapsto K(\nu^*xvec)\langle N \rangle \prec R'$
by(*metis statEqTransition Associativity*)
moreover note $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec (\nu^*xvec)\langle (P \parallel Q') \parallel R' \rangle$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_R \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_R \#* P \rangle$
 $\langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_R \#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_R \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* K \rangle \langle A_R \#* A_P \rangle \langle A_Q \#* A_R \rangle \langle xvec \#* P \rangle$
 $\langle xvec \#* Q \rangle$
by(*intro Comm1*) (*assumption* | *simp*)
moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$ **have** $(\Psi, (\nu^*xvec)\langle (P \parallel Q') \parallel R' \rangle, P \parallel ((\nu^*xvec)\langle Q' \parallel R' \rangle)) \in Rel$
by(*rule C2*)
ultimately show *?case by blast*
next
case(*cComm2* $\Psi_R M xvec N Q' A_Q \Psi_Q K R' A_R$)
from $\langle A_Q \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$
have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* R$ **and** $A_Q \#* A_P$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* \Psi_P$ **by** *simp*
from $\langle A_R \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* Q$ **and** $A_R \#* R$ **and** $A_R \#* A_P$ **and** $A_R \#* \Psi$ **by** *simp*
from $\langle xvec \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $xvec \#* A_P$ **and** $xvec \#* P$ **and** $xvec \#* Q$ **and** $xvec \#* \Psi$ **by** *simp*

have $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
have $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with $\langle A_P \#* R \rangle \langle A_R \#* A_P \rangle$ **have** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto M(\nu^*xvec)\langle N \rangle \prec Q' \rangle \langle A_P \#* Q \rangle \langle xvec \#* A_P \rangle \langle xvec \#* M \rangle \langle distinct\ xvec \rangle$ **have** $A_P \#* N$
by(*auto intro: outputFreshChainDerivative*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto M(\nu^*xvec)\langle N \rangle \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto M(\nu^*xvec)\langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto M(\nu^*xvec)\langle N \rangle \prec (P \parallel Q')$ **using** $FrP \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle \langle xvec \#* P \rangle \langle xvec \#* A_P \rangle$
by(*intro Par2*) *auto*
moreover from $FrP FrQ \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* A_P \rangle \langle A_Q \#* \Psi_P \rangle$ **have** $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto K(\nu^*xvec)\langle N \rangle \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \triangleright R \mapsto K(\nu^*xvec)\langle N \rangle \prec R'$

by(*metis statEqTransition Associativity*)
moreover note $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q)$
 $\otimes \Psi_R \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec (\nu * \text{xvec})((P \parallel Q') \parallel R')$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_R \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_R \#* P \rangle$
 $\langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_R \#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_R \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* K \rangle \langle A_R \#* A_P \rangle \langle A_Q \#* A_R \rangle \langle \text{xvec} \#* R \rangle$
by(*intro Comm2*) (*assumption* | *simp*)
moreover from $\langle \text{xvec} \#* \Psi \rangle \langle \text{xvec} \#* P \rangle$ **have** $(\Psi, (\nu * \text{xvec})((P \parallel Q') \parallel R'),$
 $P \parallel ((\nu * \text{xvec})(Q' \parallel R')) \in \text{Rel}$
by(*rule C2*)
ultimately show ?*case* **by** *blast*
next
case(*cBrMerge* $\Psi_R M N Q' A_Q \Psi_Q R' A_R$)
from $\langle A_P \#* \alpha \rangle \langle \alpha = \iota M(\downarrow N) \rangle$ **have** $A_P \#* M$ **and** $A_P \#* N$ **by** *simp*
from $\langle A_Q \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$
have $A_Q \#* P$ **and** $A_Q \#* Q$ **and** $A_Q \#* R$ **and** $A_Q \#* A_P$ **and** $A_Q \#* \Psi_P$
and $A_Q \#* \Psi$ **by** *simp*
from $\langle A_R \#* (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \#* P$ **and** $A_R \#* Q$ **and** A_R
 $\#* R$ **and** $A_R \#* A_P$ **and** $A_R \#* \Psi$ **by** *simp*

have *FrQ*: $\text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with $\langle A_P \#* Q \rangle \langle A_Q \#* A_P \rangle$ **have** $A_P \#* \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
have *FrR*: $\text{extractFrame } R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with $\langle A_P \#* R \rangle \langle A_R \#* A_P \rangle$ **have** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto \iota M(\downarrow N) \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q$
 $\mapsto \iota M(\downarrow N) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \iota M(\downarrow N) \prec (P \parallel Q')$ **using** *FrP* $\langle A_P \#* \Psi \rangle$
 $\langle A_P \#* \Psi_R \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* N \rangle$
by(*intro Par2*) *auto*
moreover from *FrP* *FrQ* $\langle A_P \#* \Psi_Q \rangle \langle A_Q \#* A_P \rangle \langle A_Q \#* \Psi_P \rangle$ **have**
 $\text{extractFrame}(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto \iota M(\downarrow N) \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes$
 $\Psi_Q \triangleright R \mapsto \iota M(\downarrow N) \prec R'$
by(*metis statEqTransition Associativity*)
moreover note $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \iota M(\downarrow N) \prec (P \parallel Q') \parallel R'$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_R \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_R \#* P \rangle$
 $\langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_R \#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_R \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_R \#* M \rangle \langle A_R \#* A_P \rangle \langle A_Q \#* A_R \rangle$
by(*auto intro: BrMerge*)
moreover have $(\Psi, (P \parallel Q') \parallel R', P \parallel (Q' \parallel R')) \in \text{Rel}$

```

    by(rule C1)
  ultimately show ?case by blast
next
case(cBrComm1  $\Psi_R M N Q' A_Q \Psi_Q xvec R' A_R$ )
from  $\langle \downarrow M(\nu * xvec) \rangle \langle N \rangle = \alpha$  have  $xvec = bn \ \alpha$ 
  by(auto simp add: action.inject)
from  $\langle \downarrow M(\nu * xvec) \rangle \langle N \rangle = \alpha$   $\langle A_P \# \alpha \rangle$ 
have  $A_P \# xvec$  and  $A_P \# M$  and  $A_P \# N$  by auto
from  $\langle xvec = bn \ \alpha \rangle \langle bn \ \alpha \# P \rangle \langle bn \ \alpha \# \Psi \rangle \langle bn \ \alpha \# \Psi_P \rangle$  have  $xvec \# P$ 
and  $xvec \# \Psi$  and  $xvec \# \Psi_P$ 
  by simp+
from  $\langle A_Q \# (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ 
have  $A_Q \# P$  and  $A_Q \# Q$  and  $A_Q \# R$  and  $A_Q \# A_P$  and  $A_Q \# \Psi_P$ 
and  $A_Q \# \Psi$  by simp+
from  $\langle A_R \# (A_P, \Psi_P, P, Q, R, \Psi) \rangle$  have  $A_R \# P$  and  $A_R \# Q$  and  $A_R$ 
 $\# R$  and  $A_R \# A_P$  and  $A_R \# \Psi$  by simp+

  have FrQ:  $extractFrame \ Q = \langle A_Q, \Psi_Q \rangle$  by fact
  with  $\langle A_P \# Q \rangle \langle A_Q \# A_P \rangle$  have  $A_P \# \Psi_Q$ 
    by(auto dest: extractFrameFreshChain)
  have FrR:  $extractFrame \ R = \langle A_R, \Psi_R \rangle$  by fact
  with  $\langle A_P \# R \rangle \langle A_R \# A_P \rangle$  have  $A_P \# \Psi_R$ 
    by(auto dest: extractFrameFreshChain)

  from  $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto \downarrow M(N) \prec Q' \rangle$  have  $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q$ 
 $\mapsto \downarrow M(N) \prec Q'$ 
    by(metis statEqTransition Associativity Commutativity Composition)
  then have  $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \downarrow M(N) \prec (P \parallel Q')$  using FrP  $\langle A_P \# \Psi \rangle$ 
 $\langle A_P \# \Psi_R \rangle \langle A_P \# Q \rangle \langle A_P \# M \rangle \langle A_P \# N \rangle$ 
    by(intro Par2) auto
  moreover from FrP FrQ  $\langle A_P \# \Psi_Q \rangle \langle A_Q \# A_P \rangle \langle A_Q \# \Psi_P \rangle$  have
 $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ 
    by simp
  moreover from  $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto \downarrow M(\nu * xvec) \rangle \langle N \rangle \prec R'$  have  $\Psi$ 
 $\otimes \Psi_P \otimes \Psi_Q \triangleright R \mapsto \downarrow M(\nu * xvec) \rangle \langle N \rangle \prec R'$ 
    by(metis statEqTransition Associativity)
  moreover note  $\langle extractFrame \ R = \langle A_R, \Psi_R \rangle \rangle$ 
  ultimately have  $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \downarrow M(\nu * xvec) \rangle \langle N \rangle \prec ((P \parallel Q') \parallel R')$ 
    using  $\langle A_P \# \Psi \rangle \langle A_Q \# \Psi \rangle \langle A_R \# \Psi \rangle \langle A_P \# P \rangle \langle A_Q \# P \rangle \langle A_R \# P \rangle$ 
 $\langle A_P \# Q \rangle \langle A_Q \# Q \rangle \langle A_R \# Q \rangle \langle A_P \# R \rangle \langle A_Q \# R \rangle \langle A_R \# R \rangle$ 
 $\langle A_P \# M \rangle \langle A_Q \# M \rangle \langle A_R \# M \rangle \langle A_R \# A_P \rangle \langle A_Q \# A_R \rangle \langle xvec \# P \rangle$ 
 $\langle xvec \# Q \rangle$ 
    by(intro BrComm1) (assumption | simp)+
  moreover have  $(\Psi, ((P \parallel Q') \parallel R'), (P \parallel (Q' \parallel R'))) \in Rel$ 
    by(rule C1)
  ultimately show ?case by blast
next
case(cBrComm2  $\Psi_R M xvec N Q' A_Q \Psi_Q R' A_R$ )
from  $\langle \downarrow M(\nu * xvec) \rangle \langle N \rangle = \alpha$  have  $xvec = bn \ \alpha$ 

```

by(*auto simp add: action.inject*)
from $\langle \text{iM}(\nu * \text{xvec}) \rangle \langle N \rangle = \alpha \rangle \langle A_P \# * \alpha \rangle$
have $A_P \# * \text{xvec}$ **and** $A_P \# * M$ **and** $A_P \# * N$ **by** *auto*
from $\langle \text{xvec} = \text{bn } \alpha \rangle \langle \text{bn } \alpha \# * P \rangle \langle \text{bn } \alpha \# * \Psi \rangle \langle \text{bn } \alpha \# * \Psi_P \rangle$ **have** $\text{xvec} \# * P$
and $\text{xvec} \# * \Psi$ **and** $\text{xvec} \# * \Psi_P$
by *simp+*
from $\langle A_Q \# * (A_P, \Psi_P, P, Q, R, \Psi) \rangle$
have $A_Q \# * P$ **and** $A_Q \# * Q$ **and** $A_Q \# * R$ **and** $A_Q \# * A_P$ **and** $A_Q \# * \Psi$ **and**
 $A_Q \# * \Psi_P$ **by** *simp+*
from $\langle A_R \# * (A_P, \Psi_P, P, Q, R, \Psi) \rangle$ **have** $A_R \# * P$ **and** $A_R \# * Q$ **and** A_R
 $\# * R$ **and** $A_R \# * A_P$ **and** $A_R \# * \Psi$ **by** *simp+*

have $\text{FrQ}: \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*
with $\langle A_P \# * Q \rangle \langle A_Q \# * A_P \rangle$ **have** $A_P \# * \Psi_Q$
by(*auto dest: extractFrameFreshChain*)
have $\text{FrR}: \text{extractFrame } R = \langle A_R, \Psi_R \rangle$ **by** *fact*
with $\langle A_P \# * R \rangle \langle A_R \# * A_P \rangle$ **have** $A_P \# * \Psi_R$
by(*auto dest: extractFrameFreshChain*)

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto \text{iM}(\nu * \text{xvec}) \rangle \langle N \rangle \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes$
 $\Psi_P \triangleright Q \mapsto \text{iM}(\nu * \text{xvec}) \rangle \langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
then **have** $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \text{iM}(\nu * \text{xvec}) \rangle \langle N \rangle \prec (P \parallel Q')$ **using** $\text{FrP} \langle A_P$
 $\# * \Psi \rangle \langle A_P \# * \Psi_R \rangle \langle A_P \# * Q \rangle \langle A_P \# * M \rangle \langle A_P \# * N \rangle \langle \text{xvec} \# * P \rangle \langle A_P \# * \text{xvec} \rangle$
by(*intro Par2*) *auto*
moreover **from** $\text{FrP} \text{FrQ} \langle A_P \# * \Psi_Q \rangle \langle A_Q \# * A_P \rangle \langle A_Q \# * \Psi_P \rangle$ **have**
 $\text{extractFrame}(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$
by *simp+*
moreover **from** $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto \text{iM}(N) \rangle \prec R' \rangle$ **have** $\Psi \otimes \Psi_P \otimes$
 $\Psi_Q \triangleright R \mapsto \text{iM}(N) \rangle \prec R'$
by(*metis statEqTransition Associativity*)
moreover **note** $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
ultimately **have** $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \text{iM}(\nu * \text{xvec}) \rangle \langle N \rangle \prec ((P \parallel Q') \parallel R')$
using $\langle A_P \# * \Psi \rangle \langle A_Q \# * \Psi \rangle \langle A_R \# * \Psi \rangle \langle A_P \# * P \rangle \langle A_Q \# * P \rangle \langle A_R \# * P \rangle$
 $\langle A_P \# * Q \rangle \langle A_Q \# * Q \rangle \langle A_R \# * Q \rangle \langle A_P \# * R \rangle \langle A_Q \# * R \rangle \langle A_R \# * R \rangle$
 $\langle A_P \# * M \rangle \langle A_Q \# * M \rangle \langle A_R \# * M \rangle \langle A_R \# * A_P \rangle \langle A_Q \# * A_R \rangle \langle \text{xvec} \# * R \rangle$
by(*auto intro: BrComm2*)
moreover **have** $(\Psi, ((P \parallel Q') \parallel R'), (P \parallel (Q' \parallel R'))) \in \text{Rel}$
by(*rule C1*)
ultimately **show** *?case* **by** *blast*

qed
next
case(*cComm1* $\Psi_{QR} M N P' A_P \Psi_P K \text{xvec} QR' A_{QR}$)
from $\langle \text{xvec} \# * (\Psi, P, Q, R, \alpha) \rangle$ **have** $\text{xvec} \# * Q$ **and** $\text{xvec} \# * R$ **by** *simp+*
from $\langle A_{QR} \# * (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \# * Q$ **and** $A_{QR} \# * R$ **and** $A_{QR} \# *$
 Ψ **by** *simp+*
from $\langle A_P \# * (Q \parallel R) \rangle$ **have** $A_P \# * Q$ **and** $A_P \# * R$ **by** *simp+*
have $P\text{Trans}: \Psi \otimes \Psi_{QR} \triangleright P \mapsto M(N) \rangle \prec P'$ **and** $\text{FrP}: \text{extractFrame } P =$
 $\langle A_P, \Psi_P \rangle$ **and** $\text{MeqK}: \Psi \otimes \Psi_P \otimes \Psi_{QR} \vdash M \leftrightarrow K$ **by** *fact+*

note $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto K(\nu * xvec)\langle N \rangle \prec QR' \rangle$
moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ **have** $xvec \#* (\Psi \otimes \Psi_P)$ **by force**
moreover note $\langle xvec \#* Q \rangle \langle xvec \#* R \rangle \langle xvec \#* K \rangle$
 $\langle extractFrame(Q \parallel R) = \langle A_{QR}, \Psi_{QR} \rangle \rangle \langle distinct A_{QR} \rangle$
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle$ **have** $A_{QR} \#* (\Psi \otimes \Psi_P)$ **by force**
ultimately show *?case using* $\langle A_{QR} \#* Q \rangle \langle A_{QR} \#* R \rangle \langle A_{QR} \#* K \rangle$
proof(*induct rule: parCasesOutputFrame*)
case(*cPar1 Q' A_Q Ψ_Q A_R Ψ_R*)
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by fact+**
from *PTrans Ψeq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto M(\langle N \rangle) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto K(\nu * xvec)\langle N \rangle \prec Q' \rangle$ **have** $(\Psi$
 $\otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec)\langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from *MeqK Ψeq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
by(*metis statEqEnt Associativity Commutativity Composition*)
moreover from $\langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq* $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi \rangle$ *Aeq* **have** $A_Q \#* P$ **and** $A_Q \#* \Psi$
by simp+
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \tau \prec (\nu * xvec)\langle P' \parallel Q' \rangle$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle xvec \#* P \rangle$
by(*intro Comm1*) (*assumption | force*)
moreover from $\langle A_{QR} \#* \Psi \rangle$ *Aeq* **have** $A_R \#* \Psi$ **by simp**
moreover from $\langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_R \#* P$ **by simp**
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec ((\nu * xvec)\langle P' \parallel Q' \rangle) \parallel R$ **using**
 $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* Q \rangle$
by(*intro Par1*) (*assumption | simp*)
moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* R \rangle$ **have** $(\Psi, ((\nu * xvec)\langle P' \parallel Q' \rangle) \parallel R,$
 $(\nu * xvec)\langle P' \parallel (Q' \parallel R) \rangle) \in Rel$
by(*rule C3*)
ultimately show *?case by blast*
next
case(*cPar2 R' A_Q Ψ_Q A_R Ψ_R*)
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by fact+**
from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi_P \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq* **have** $A_R \#*$
 Ψ **and** $A_R \#* \Psi_P$ **and** $A_P \#* A_R$ **and** $A_P \#* A_Q$ **and** $A_R \#* P$ **by simp+**
from $\langle A_{QR} \#* \Psi \rangle$ *Aeq* **have** $A_Q \#* \Psi$ **by simp**
from $\langle A_{QR} \#* P \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq* *FrP* **have** $A_Q \#* \Psi_P$ **by**(*auto dest:*
extractFrameFreshChain)
from $\langle A_P \#* A_{QR} \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle extractFrame Q = \langle A_Q,$
 $\Psi_Q \rangle \rangle$ *Aeq* $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle$ **have** $A_P \#* \Psi_Q$ **and** $A_P \#* \Psi_R$ **by**(*auto dest:*
extractFrameFreshChain)
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto K(\nu * xvec)\langle N \rangle \prec R'$ **and** *FrR*:
 $extractFrame R = \langle A_R, \Psi_R \rangle$ **by fact+**

then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto R \text{Out } K \ (\langle \nu^* \text{vec} \rangle N \prec' R')$ **by** (*simp add: residualInject*)

then obtain K' **where** $\text{Keq}K'$: $((\Psi \otimes \Psi_P) \otimes \Psi_Q) \otimes \Psi_R \vdash K \leftrightarrow K'$ **and** $A_P \#* K'$ **and** $A_Q \#* K'$

using $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_Q \#* A_R \rangle \langle A_P \#* A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* R \rangle \langle A_R \#* K \rangle \langle \text{distinct } A_R \rangle \langle \text{vec } \#* K \rangle \langle \text{distinct } \text{vec} \rangle \text{FrR}$

using *outputObtainPrefix* [**where** $B = A_P \text{@} A_Q$]

by (*smt (verit, ccfv-threshold) freshChainAppend freshChainSym freshCompChain(1)*)

from $P \text{Trans } \Psi \text{eq}$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto M(\langle N \rangle) \prec P'$

by (*metis statEqTransition Associativity Commutativity Composition*)

moreover from $\text{Meq}K \ \text{Keq}K' \ \Psi \text{eq}$ **have** $\text{Meq}K'$: $((\Psi \otimes \Psi_R) \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$

by (*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)

ultimately have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto K'(\langle N \rangle) \prec P'$ **using** $\text{FrP} \langle \text{distinct } A_P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K' \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* \Psi_R \rangle$

by (*auto intro: inputRenameSubject*)

moreover from $\langle A_{QR} \#* P \rangle \langle A_{QR} \#* N \rangle \text{Aeq}$ **have** $A_Q \#* P$ **and** $A_Q \#* N$

by *simp+*

ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto K'(\langle N \rangle) \prec P' \parallel Q$ **using** $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi_R \rangle \langle A_Q \#* K' \rangle \langle A_Q \#* \Psi \rangle$

by (*intro Par1 (assumption | force)+*)

moreover from $\text{FrP} \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$

have $\text{extractFrame}(P \parallel Q) = \langle (A_P \text{@} A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*

moreover from $R \text{Trans}$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto K(\langle \nu^* \text{vec} \rangle \langle N \rangle) \prec R'$ **by** (*metis Associativity statEqTransition*)

moreover note FrR

moreover from $\text{Meq}K' \ \text{Keq}K'$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash K' \leftrightarrow K$

by (*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)

ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec (\langle \nu^* \text{vec} \rangle ((P' \parallel Q) \parallel R'))$

using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* K' \rangle \langle A_Q \#* K' \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* K \rangle \langle \text{vec } \#* P \rangle \langle \text{vec } \#* Q \rangle$

by (*intro Comm1 (assumption | simp)+*)

moreover from $\langle \text{vec } \#* \Psi \rangle$ **have** $(\Psi, (\langle \nu^* \text{vec} \rangle ((P' \parallel Q) \parallel R'), (\langle \nu^* \text{vec} \rangle (P' \parallel (Q \parallel R')))) \in \text{Rel}$

by (*metis C1 C4*)

ultimately show *?case by blast*

qed

next

case (*cComm2* $\Psi_{QR} \ M \ \text{vec} \ N \ P' \ A_P \ \Psi_P \ K \ QR' \ A_{QR}$)

from $\langle A_{QR} \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \#* Q$ **and** $A_{QR} \#* R$ **and** $A_{QR} \#* \Psi$ **by** *simp+*

from $\langle A_P \#* (Q \parallel R) \rangle \langle \text{vec } \#* (Q \parallel R) \rangle$ **have** $A_P \#* Q$ **and** $A_P \#* R$ **and** $\text{vec } \#* Q$ **and** $\text{vec } \#* R$ **by** *simp+*

have $PTrans: \Psi \otimes \Psi_{QR} \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$ **and** $FrP: extractFrame$
 $P = \langle A_P, \Psi_P \rangle$ **and** $MeqK: \Psi \otimes \Psi_P \otimes \Psi_{QR} \vdash M \leftrightarrow K$ **by** $fact+$
note $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto K(\lfloor N \rfloor) \prec QR' \rangle \langle extractFrame(Q \parallel R) = \langle A_{QR},$
 $\Psi_{QR} \rangle \rangle \langle distinct A_{QR} \rangle$
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle$ **have** $A_{QR} \#* (\Psi \otimes \Psi_P)$ **by** $force$
ultimately show $?case$ **using** $\langle A_{QR} \#* Q \rangle \langle A_{QR} \#* R \rangle \langle A_{QR} \#* K \rangle$
proof($induct$ rule: $parCasesInputFrame$)
case($cPar1$ $Q' A_Q \Psi_Q A_R \Psi_R$)
have $Aeq: A_{QR} = A_Q @ A_R$ **and** $\Psi eq: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** $fact+$
from $PTrans \Psi eq$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P'$
by($metis$ $statEqTransition$ $Associativity$ $Commutativity$ $Composition$)
moreover note FrP
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto K(\lfloor N \rfloor) \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes$
 $\Psi_P \triangleright Q \mapsto K(\lfloor N \rfloor) \prec Q'$
by($metis$ $statEqTransition$ $Associativity$ $Commutativity$ $Composition$)
moreover note $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $MeqK \Psi eq$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
by($metis$ $statEqEnt$ $Associativity$ $Commutativity$ $Composition$)
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ Aeq $\langle extractFrame Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by**($auto$ $dest: extractFrameFreshChain$)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ Aeq **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by $simp+$
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \tau \prec (\nu*xvec)(P' \parallel Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle xvec \#* Q \rangle$
by($intro$ $Comm2$) ($assumption$ | $force$)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ Aeq **have** $A_R \#* \Psi$ **and** $A_R \#* P$
by $simp+$
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec ((\nu*xvec)(P' \parallel Q')) \parallel R$ **using**
 $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* Q \rangle$
by($intro$ $Par1$) ($assumption$ | $simp$)
moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* R \rangle$ **have** $(\Psi, ((\nu*xvec)(P' \parallel Q')) \parallel R,$
 $(\nu*xvec)(P' \parallel (Q' \parallel R))) \in Rel$
by($rule$ $C3$)
ultimately show $?case$ **by** $blast$
next
case($cPar2$ $R' A_Q \Psi_Q A_R \Psi_R$)
have $Aeq: A_{QR} = A_Q @ A_R$ **and** $\Psi eq: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** $fact+$
from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi_P \rangle \langle A_P \#* A_{QR} \rangle$ Aeq
have $A_R \#* \Psi$ **and** $A_R \#* \Psi_P$ **and** $A_P \#* A_R$ **and** $A_P \#* A_Q$ **and** $A_R \#* P$
and $A_Q \#* \Psi$ **and** $A_Q \#* \Psi_P$ **by** $simp+$
have $RTrans: (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto K(\lfloor N \rfloor) \prec R'$ **and** $FrR: extractFrame$
 $R = \langle A_R, \Psi_R \rangle$ **by** $fact+$
then obtain K' **where** $KeqK': ((\Psi \otimes \Psi_P) \otimes \Psi_Q) \otimes \Psi_R \vdash K \leftrightarrow K'$ **and**
 $A_P \#* K'$ **and** $A_Q \#* K'$
using $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle \langle A_Q \#*$
 $A_R \rangle \langle A_P \#* A_R \rangle \langle A_R \#* R \rangle \langle A_R \#* R \rangle \langle A_R \#* K \rangle \langle distinct A_R \rangle$
using $inputObtainPrefix$ [**where** $B = A_P @ A_Q$]

by (*smt (verit, ccfv-threshold) freshChainAppend freshChainSym freshCompChain(1)*)
from $PTrans \Psi eq$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover from $MeqK KeqK' \Psi eq$ **have** $MeqK': ((\Psi \otimes \Psi_R) \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow K'$
by (*metis statEqEnt Associativity Commutativity Composition chanEqTrans*)
moreover from $\langle A_P \#* R \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_{QR} \rangle FrR \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle Aeq$ **have** $A_P \#* \Psi_Q$ **and** $A_P \#* \Psi_R$
by (*auto dest: extractFrameFreshChain*)
ultimately have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto K'(\nu * xvec)\langle N \rangle \prec P'$ **using** $FrP \langle distinct A_P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* K' \rangle$
by (*auto intro: outputRenameSubject*)
moreover from $\langle A_{QR} \#* P \rangle \langle A_{QR} \#* N \rangle \langle A_{QR} \#* xvec \rangle Aeq$ **have** $A_Q \#* P$
and $A_Q \#* N$ **and** $A_Q \#* xvec$ **by** *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto K'(\nu * xvec)\langle N \rangle \prec (P' \parallel Q)$ **using** $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi_R \rangle \langle A_Q \#* K' \rangle \langle xvec \#* Q \rangle \langle A_Q \#* \Psi \rangle$
by (*intro Par1 (assumption | force)*)
moreover from $FrP \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle$
have $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*
moreover from $RTrans$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto K(\langle N \rangle) \prec R'$ **by** (*metis Associativity statEqTransition*)
moreover note FrR
moreover from $MeqK' KeqK'$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \otimes \Psi_R \vdash K' \leftrightarrow K$
by (*metis statEqEnt Associativity Commutativity Composition chanEqTrans chanEqSym*)
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \tau \prec (\nu * xvec)((P' \parallel Q) \parallel R')$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* K' \rangle \langle A_Q \#* K' \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* K \rangle \langle xvec \#* R \rangle$
by (*intro Comm2 (assumption | simp)*)
moreover from $\langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu * xvec)((P' \parallel Q) \parallel R'), (\nu * xvec)(P' \parallel (Q \parallel R')) \in Rel$
by (*metis C1 C4*)
ultimately show *?case by blast*
qed
next
case (*cBrMerge* $\Psi_{QR} M N P' A_P \Psi_P QR' A_{QR}$)
from $\langle A_{QR} \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \#* Q$ **and** $A_{QR} \#* R$ **and** $A_{QR} \#* \Psi$ **by** *simp+*
from $\langle A_P \#* (Q \parallel R) \rangle$ **have** $A_P \#* Q$ **and** $A_P \#* R$ **by** *simp+*
have $PTrans: \Psi \otimes \Psi_{QR} \triangleright P \mapsto iM(\langle N \rangle) \prec P'$ **and** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
note $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto iM(\langle N \rangle) \prec QR' \rangle \langle extractFrame(Q \parallel R) = \langle A_{QR}, \Psi_{QR} \rangle \rangle \langle distinct A_{QR} \rangle$
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle$ **have** $A_{QR} \#* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case using* $\langle A_{QR} \#* Q \rangle \langle A_{QR} \#* R \rangle \langle A_{QR} \#* M \rangle \langle A_{QR} \#* N \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* M \rangle \langle A_{QR} \#* \Psi \rangle$

proof(*induct rule: parCasesBrInputFrame*)
case(*cPar1* $Q' A_Q \Psi_Q A_R \Psi_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *PTrans* *Ψeq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto_i M(N) \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R)$
 $\otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq* $\langle \text{extractFrame } Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by**(*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(N) \prec (P' \parallel Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
by(*intro BrMerge*) (*assumption* | *force*)+
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_R \#* \Psi$ **and** $A_R \#* P$
by *simp+*
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(N) \prec (P' \parallel Q') \parallel R$ **using**
 $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* Q \rangle \langle A_R \#* M \rangle \langle A_R \#* N \rangle$
by(*intro Par1*) (*assumption* | *simp*)+
moreover have $(\Psi, (P' \parallel Q') \parallel R, (P' \parallel (Q' \parallel R))) \in \text{Rel}$
by(*rule C1*)
ultimately show *?case by blast*
next
case(*cPar2* $R' A_Q \Psi_Q A_R \Psi_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi_P \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq*
have $A_R \#* \Psi$ **and** $A_R \#* \Psi_P$ **and** $A_P \#* A_R$ **and** $A_P \#* A_Q$ **and** $A_R \#* P$
and $A_Q \#* \Psi$ **and** $A_Q \#* \Psi_P$ **by** *simp+*
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_i M(N) \prec R'$ **and** *FrR*: extractFrame
 $R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
from *PTrans* *Ψeq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover from $\langle A_P \#* R \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_{QR} \rangle$ *FrR* $\langle \text{extractFrame } Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle$ *Aeq* **have** $A_P \#* \Psi_Q$ **and** $A_P \#* \Psi_R$
by(*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* P \rangle \langle A_{QR} \#* N \rangle$ *Aeq* **have** $A_Q \#* P$ **and** $A_Q \#* N$
by *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(N) \prec (P' \parallel Q)$ **using** $\langle \text{extract-$
 $\text{Frame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi_R \rangle \langle A_Q \#* M \rangle \langle A_Q \#* \Psi \rangle$
by(*intro Par1*) (*assumption* | *force*)+

moreover from FrP $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** $simp+$
moreover from $RTrans$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_{iM(N)} \prec R'$
by(*metis Associativity statEqTransition*)
moreover note FrR
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_{iM(N)} \prec ((P' \parallel Q) \parallel R')$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$
 $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle$
by(*auto intro: BrMerge*)
moreover have $(\Psi, ((P' \parallel Q) \parallel R'), (P' \parallel (Q \parallel R'))) \in Rel$
by(*rule C1*)
ultimately show *?case* **by** *blast*
next
case(*cBrMerge* $\Psi_R\ Q'\ A_Q\ \Psi_Q\ R'\ A_R$)
have $Aeq: A_{QR} = A_Q @ A_R$ **and** $\Psi eq: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** $fact+$
from $\langle A_{QR} \#* N \rangle \langle A_{QR} \#* M \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle \langle A_P$
 $\#* A_{QR} \rangle Aeq\ \Psi eq$
have $A_Q \#* N$ **and** $A_R \#* N$ **and** $A_Q \#* M$ **and** $A_R \#* M$
and $A_Q \#* \Psi$ **and** $A_R \#* \Psi$ **and** $A_P \#* A_Q$ **and** $A_P \#* A_R$
and $A_Q \#* \Psi_P$ **and** $A_R \#* \Psi_P$
and $A_Q \#* P$ **and** $A_R \#* P$
by $simp+$

from $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_Q \rangle$
have $A_P \#* \Psi_Q$
by (*metis extractFrameFreshChain freshFrameDest*)
from $Aeq\ \Psi eq\ PTrans$ **have** $\Psi \otimes (\Psi_Q \otimes \Psi_R) \triangleright P \mapsto_{iM(N)} \prec P'$ **by** $simp$
then have $\Psi \otimes (\Psi_R \otimes \Psi_Q) \triangleright P \mapsto_{iM(N)} \prec P'$
by (*metis Commutativity* Ψeq *compositionSym statEqTransition*)
then have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_{iM(N)} \prec P'$
by (*metis AssertionStatEqSym Associativity statEqTransition*)
moreover note FrP
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto_{iM(N)} \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R)$
 $\otimes \Psi_P \triangleright Q \mapsto_{iM(N)} \prec Q'$
by (*metis associativitySym statEqTransition*)
moreover note $\langle extractFrame\ Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_R \rangle$
moreover from $\langle extractFrame\ R = \langle A_R, \Psi_R \rangle \rangle \langle A_P \#* A_R \rangle \langle A_P \#* R \rangle$ **have**
 $A_P \#* \Psi_R$
by (*metis extractFrameFreshChain freshFrameDest*)
moreover note $\langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* P \rangle$
 $\langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
ultimately have $(\Psi \otimes \Psi_R) \triangleright (P \parallel Q) \mapsto_{iM(N)} \prec (P' \parallel Q')$
by(*intro BrMerge*) (*assumption* | *force*)+

from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_{iM(N)} \prec R' \rangle$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R$
 $\mapsto_{iM(N)} \prec R'$

by (*metis Associativity statEqTransition*)

note $\langle (\Psi \otimes \Psi_R) \triangleright (P \parallel Q) \mapsto_i M(N) \prec (P' \parallel Q') \rangle$
moreover from $FrP \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), (\Psi_P \otimes \Psi_Q) \rangle$ **by** *simp*
moreover note $\langle \Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_i M(N) \prec R' \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_R \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$
 $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle$
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(N) \prec (P' \parallel Q') \parallel R'$
by (*auto intro: BrMerge*)
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by (*rule C1*)
ultimately show *?case by blast*

qed

next

case (*cBrComm1* $\Psi_{QR} M N P' A_P \Psi_P xvec QR' A_{QR}$)
from $\langle_i M(\nu * xvec) \langle N \rangle = \alpha \rangle \langle bn \alpha \#* Q \rangle \langle bn \alpha \#* R \rangle$ **have** $xvec \#* Q$ **and** $xvec \#* R$ **by** *auto*
from $\langle A_{QR} \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \#* Q$ **and** $A_{QR} \#* R$ **and** $A_{QR} \#* \Psi$ **by** *simp+*
from $\langle A_P \#* (Q \parallel R) \rangle$ **have** $A_P \#* Q$ **and** $A_P \#* R$ **by** *simp+*
have $PTrans: \Psi \otimes \Psi_{QR} \triangleright P \mapsto_i M(N) \prec P'$ **and** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
note $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto_i M(\nu * xvec) \langle N \rangle \prec QR' \rangle$
moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ **have** $xvec \#* (\Psi \otimes \Psi_P)$ **by** *force*
moreover note $\langle xvec \#* Q \rangle \langle xvec \#* R \rangle \langle xvec \#* M \rangle$
 $\langle extractFrame(Q \parallel R) = \langle A_{QR}, \Psi_{QR} \rangle \rangle \langle distinct A_{QR} \rangle$
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle$ **have** $A_{QR} \#* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case using* $\langle A_{QR} \#* Q \rangle \langle A_{QR} \#* R \rangle \langle A_{QR} \#* M \rangle$
proof (*induct rule: parCasesBrOutputFrame*)
case (*cPar1* $Q' A_Q \Psi_Q A_R \Psi_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
from $\langle A_{QR} \#* xvec \rangle \langle A_{QR} = (A_Q @ A_R) \rangle$ **have** $A_Q \#* xvec$ **and** $A_R \#* xvec$
by *simp+*
have $Aeq: A_{QR} = A_Q @ A_R$ **and** $\Psi eq: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from $PTrans \Psi eq$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_i M(N) \prec P'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover note FrP
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec Q'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ $Aeq \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$

have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by** (*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec (P' \parallel Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle xvec \#* P \rangle$
by (*intro BrComm1*) (*assumption* | *force*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_R \#* \Psi$ **and** $A_R \#* P$
by *simp+*
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(\nu * xvec) \langle N \rangle \prec (P' \parallel Q') \parallel R$
using $\langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* Q \rangle \langle A_R \#* M \rangle \langle A_R \#* N \rangle \langle xvec \#* R \rangle$
 $\langle A_R \#* xvec \rangle$
by (*intro Par1*) (*assumption* | *simp*)
moreover have $(\Psi, (P' \parallel Q') \parallel R, (P' \parallel (Q' \parallel R))) \in Rel$
by (*rule C1*)
ultimately show *?case by blast*
next
case (*cPar2 R' A_Q \Psi_Q A_R \Psi_R*)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *\Psieq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \langle A_{QR} \#* \Psi_P \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq*
have $A_R \#* \Psi$ **and** $A_R \#* \Psi_P$ **and** $A_P \#* A_R$ **and** $A_P \#* A_Q$ **and** $A_R \#* P$
and $A_Q \#* \Psi$ **and** $A_Q \#* \Psi_P$ **by** *simp+*
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_i M(\nu * xvec) \langle N \rangle \prec R'$ **and** *FrR*:
 $extractFrame R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
from *PTrans \Psieq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * xvec) \langle N \rangle \prec P'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover from $\langle A_P \#* R \rangle \langle A_P \#* Q \rangle \langle A_P \#* A_{QR} \rangle$ *FrR* $\langle extractFrame Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle$ *Aeq* **have** $A_P \#* \Psi_Q$ **and** $A_P \#* \Psi_R$
by (*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* P \rangle \langle A_{QR} \#* N \rangle$ *Aeq* **have** $A_Q \#* P$ **and** $A_Q \#* N$
by *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(\nu * xvec) \langle N \rangle \prec (P' \parallel Q)$ **using** $\langle extract-$
 $Frame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \#* \Psi_R \rangle \langle A_Q \#* M \rangle \langle A_Q \#* \Psi \rangle$
by (*intro Par1*) (*assumption* | *force*)
moreover from *FrP* $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_Q \#* \Psi_P \rangle$
have $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_i M(\nu * xvec) \langle N \rangle \prec$
 R' **by** (*metis Associativity statEqTransition*)
moreover note *FrR*
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(\nu * xvec) \langle N \rangle \prec ((P' \parallel Q) \parallel R')$
using $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_Q \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* Q \rangle$
 $\langle A_P \#* R \rangle \langle A_Q \#* R \rangle \langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$
 $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle xvec \#* P \rangle \langle xvec$
 $\#* Q \rangle$
by (*intro BrComm1*) (*assumption* | *simp*)
moreover have $(\Psi, ((P' \parallel Q) \parallel R), (P' \parallel (Q \parallel R))) \in Rel$

by(rule C1)
 ultimately show ?case by blast
 next
 case(cBrComm1 $\Psi_R Q' A_Q \Psi_Q R' A_R$)
 from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ have $A_Q \#* N$ and $A_R \#* N$
 by simp+
 from $\langle A_{QR} \#* xvec \rangle \langle A_{QR} = (A_Q @ A_R) \rangle$ have $A_Q \#* xvec$ and $A_R \#* xvec$
 by simp+
 have Aeq: $A_{QR} = A_Q @ A_R$ and $\Psi eq: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ by fact+
 from Aeq $\langle A_{QR} \#* P \rangle$ have $A_Q \#* P$ and $A_R \#* P$ by simp+
 from Aeq $\langle A_{QR} \#* \Psi \rangle$ have $A_Q \#* \Psi$ and $A_R \#* \Psi$ by simp+
 from Aeq $\langle A_P \#* A_{QR} \rangle$ have $A_P \#* A_Q$ and $A_P \#* A_R$ by simp+
 with $\langle A_P \#* Q \rangle \langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$ have $A_P \#* \Psi_Q$
 by (metis extractFrameFreshChain freshChainSym freshFrameDest)
 from PTrans Ψeq have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_i M(|N|) \prec P'$
 by (metis statEqTransition Associativity Commutativity Composition)
 moreover note FrP
 moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto_i M(|N|) \prec Q' \rangle$ have $(\Psi \otimes \Psi_R)$
 $\otimes \Psi_P \triangleright Q \mapsto_i M(|N|) \prec Q'$
 by (metis statEqTransition Associativity Commutativity Composition)
 moreover note $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle$
 moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ Aeq $\langle extractFrame Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle \langle extractFrame R = \langle A_R, \Psi_R \rangle \rangle$
 have $A_P \#* A_Q$ and $A_P \#* \Psi_R$ by (auto dest: extractFrameFreshChain)
 moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ Aeq have $A_Q \#* \Psi$ and $A_Q \#* P$
 by simp+
 ultimately have PQTrans: $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(|N|) \prec (P' \parallel Q')$
 using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle$
 by (intro BrMerge) (assumption | force)+
 have RTrans: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_i M(|\nu * xvec|)(N) \prec R'$ and FrR:
 $extractFrame R = \langle A_R, \Psi_R \rangle$ by fact+
 note PQTrans
 moreover from FrP $\langle extractFrame Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_{QR} \#* \Psi_P \rangle$ Aeq
 have $extractFrame(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ by simp+
 moreover from RTrans have $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_i M(|\nu * xvec|)(N) \prec$
 R'
 by (metis Associativity statEqTransition)
 moreover note FrR
 moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* P \rangle \langle A_Q$
 $\#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R$
 $\#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle$
 ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(|\nu * xvec|)(N) \prec (P' \parallel Q') \parallel R'$
 by (intro BrComm1) (assumption | force)+
 moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
 by (rule C1)
 ultimately show ?case by blast

next
case(*cBrComm2* $\Psi_R Q' A_Q \Psi_Q R' A_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
from $\langle A_{QR} \#* xvec \rangle \langle A_{QR} = (A_Q @ A_R) \rangle$ **have** $A_Q \#* xvec$ **and** $A_R \#* xvec$
by *simp+*
have *Aeq*: $A_{QR} = A_Q @ A_R$ **and** *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from *Aeq* $\langle A_{QR} \#* P \rangle$ **have** $A_Q \#* P$ **and** $A_R \#* P$ **by** *simp+*
from *Aeq* $\langle A_{QR} \#* \Psi \rangle$ **have** $A_Q \#* \Psi$ **and** $A_R \#* \Psi$ **by** *simp+*
from *Aeq* $\langle A_P \#* A_{QR} \rangle$ **have** $A_P \#* A_Q$ **and** $A_P \#* A_R$ **by** *simp+*
with $\langle A_P \#* Q \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $A_P \#* \Psi_Q$
by (*metis extractFrameFreshChain freshChainSym freshFrameDest*)
from *PTrans* *Ψeq* **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_{iM} \langle N \rangle \prec P'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover note *FrP*
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto_{iM} \langle \nu * xvec \rangle \langle N \rangle \prec Q' \rangle$ **have** $(\Psi$
 $\otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto_{iM} \langle \nu * xvec \rangle \langle N \rangle \prec Q'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ *Aeq* $\langle \text{extractFrame } Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by** (*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ *Aeq* **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by *simp+*
ultimately have *PQTrans*: $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_{iM} \langle \nu * xvec \rangle \langle N \rangle \prec (P' \parallel$
 $Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle xvec \#* P \rangle$
by (*intro BrComm1*) (*assumption* | *force*)
have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_{iM} \langle N \rangle \prec R'$ **and** *FrR*: extractFrame
 $R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
note *PQTrans*
moreover from *FrP* $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_{QR} \#* \Psi_P \rangle$ *Aeq*
have $\text{extractFrame}(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*
moreover from *RTrans* **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_{iM} \langle N \rangle \prec R'$
by (*metis Associativity statEqTransition*)
moreover note *FrR*
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* P \rangle \langle A_Q$
 $\#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R$
 $\#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle$
 $\langle xvec \#* P \rangle \langle xvec \#* Q \rangle \langle xvec \#* R \rangle$
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_{iM} \langle \nu * xvec \rangle \langle N \rangle \prec (P' \parallel Q') \parallel R'$
by (*auto intro: BrComm2*)
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R')))) \in \text{Rel}$
by (*rule C1*)
ultimately show *?case* **by** *blast*
qed

next
case(*cBrComm2* $\Psi_{QR} M \text{vec} N P' A_P \Psi_P QR' A_{QR}$)
from $\langle \text{iM}(\nu * \text{vec}) \langle N \rangle = \alpha \rangle \langle \text{bn } \alpha \#* Q \rangle \langle \text{bn } \alpha \#* R \rangle$ **have** $\text{vec} \#* Q$ **and** $\text{vec} \#* R$ **by** *auto*
from $\langle A_{QR} \#* (\Psi, P, Q, R, \alpha) \rangle$ **have** $A_{QR} \#* Q$ **and** $A_{QR} \#* R$ **and** $A_{QR} \#* \Psi$ **by** *simp+*
from $\langle A_P \#* (Q \parallel R) \rangle$ **have** $A_P \#* Q$ **and** $A_P \#* R$ **by** *simp+*
have $P\text{Trans}: \Psi \otimes \Psi_{QR} \triangleright P \mapsto \text{iM}(\nu * \text{vec}) \langle N \rangle \prec P'$ **and** $\text{FrP}: \text{extractFrame} P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
note $\langle \Psi \otimes \Psi_P \triangleright Q \parallel R \mapsto \text{iM} \langle N \rangle \prec QR' \rangle \langle \text{extractFrame}(Q \parallel R) = \langle A_{QR}, \Psi_{QR} \rangle \rangle \langle \text{distinct } A_{QR} \rangle$
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* \Psi_P \rangle$ **have** $A_{QR} \#* (\Psi \otimes \Psi_P)$ **by** *force*
ultimately show *?case using* $\langle A_{QR} \#* Q \rangle \langle A_{QR} \#* R \rangle \langle A_{QR} \#* M \rangle$
proof(*induct rule: parCasesBrInputFrame*)
case(*cPar1* $Q' A_Q \Psi_Q A_R \Psi_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$
by *simp+*
from $\langle A_{QR} \#* \text{vec} \rangle \langle A_{QR} = (A_Q @ A_R) \rangle$ **have** $A_Q \#* \text{vec}$ **and** $A_R \#* \text{vec}$
by *simp+*
have $\text{Aeq}: A_{QR} = A_Q @ A_R$ **and** $\Psi\text{eq}: \Psi_{QR} = \Psi_Q \otimes \Psi_R$ **by** *fact+*
from $P\text{Trans} \Psi\text{eq}$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto \text{iM}(\nu * \text{vec}) \langle N \rangle \prec P'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note FrP
moreover from $\langle (\Psi \otimes \Psi_P) \otimes \Psi_R \triangleright Q \mapsto \text{iM} \langle N \rangle \prec Q' \rangle$ **have** $(\Psi \otimes \Psi_R) \otimes \Psi_P \triangleright Q \mapsto \text{iM} \langle N \rangle \prec Q'$
by(*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle$ Aeq $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by**(*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ Aeq **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by *simp+*
ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto \text{iM}(\nu * \text{vec}) \langle N \rangle \prec (P' \parallel Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle \text{vec} \#* Q \rangle$
by(*intro BrComm2*) (*assumption | force*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle$ Aeq **have** $A_R \#* \Psi$ **and** $A_R \#* P$
by *simp+*
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto \text{iM}(\nu * \text{vec}) \langle N \rangle \prec (P' \parallel Q') \parallel R$
using $\langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle \langle A_R \#* Q \rangle \langle A_R \#* M \rangle \langle A_R \#* N \rangle \langle \text{vec} \#* R \rangle \langle A_R \#* \text{vec} \rangle$
by(*intro Par1*) (*assumption | simp*)
moreover have $(\Psi, (P' \parallel Q') \parallel R, (P' \parallel (Q' \parallel R))) \in \text{Rel}$
by(*rule C1*)
ultimately show *?case by blast*
next
case(*cPar2* $R' A_Q \Psi_Q A_R \Psi_R$)
from $\langle A_{QR} \#* N \rangle \langle A_{QR} \#* \text{vec} \rangle \langle A_{QR} = A_Q @ A_R \rangle$ **have** $A_Q \#* N$ **and** $A_R \#* N$ **and** $A_Q \#* \text{vec}$ **and** $A_R \#* \text{vec}$

by *simp+*
 have *Aeq*: $A_{QR} = A_Q @ A_R$ and *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ by *fact+*
 from $\langle A_{QR} \# \Psi \rangle \langle A_{QR} \# P \rangle \langle A_{QR} \# \Psi_P \rangle \langle A_P \# A_{QR} \rangle$ *Aeq*
 have $A_R \# \Psi$ and $A_R \# \Psi_P$ and $A_P \# A_R$ and $A_P \# A_Q$ and $A_R \# P$
 and $A_Q \# \Psi$ and $A_Q \# \Psi_P$ by *simp+*
 have *RTrans*: $(\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_{iM} (\nu N) \prec R'$ and *FrR*: *extractFrame*
 $R = \langle A_R, \Psi_R \rangle$ by *fact+*
 from *PTrans* *Ψeq* have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec P'$
 by (*metis statEqTransition Associativity Commutativity Composition*)
 moreover from $\langle A_P \# R \rangle \langle A_P \# Q \rangle \langle A_P \# A_{QR} \rangle$ *FrR* $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ *Aeq* have $A_P \# \Psi_Q$ and $A_P \# \Psi_R$
 by (*auto dest: extractFrameFreshChain*)
 moreover from $\langle A_{QR} \# P \rangle \langle A_{QR} \# N \rangle$ *Aeq* have $A_Q \# P$ and $A_Q \# N$
 by *simp+*
 ultimately have $\Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (P' \parallel Q)$ using
 $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_Q \# \Psi_R \rangle$
 $\langle A_Q \# M \rangle \langle A_Q \# \Psi \rangle \langle xvec \# Q \rangle \langle A_Q \# xvec \rangle$
 by (*intro Par1*) (*assumption* | *force*) +
 moreover from *FrP* $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \# A_Q \rangle \langle A_P \# \Psi_Q \rangle$
 $\langle A_Q \# \Psi_P \rangle$
 have *extractFrame*($P \parallel Q$) = $\langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ by *simp+*
 moreover from *RTrans* have $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_{iM} (\nu N) \prec R'$
 by (*metis Associativity statEqTransition*)
 moreover note *FrR*
 ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec ((P' \parallel Q) \parallel R')$
 using $\langle A_P \# \Psi \rangle \langle A_Q \# \Psi \rangle \langle A_P \# P \rangle \langle A_Q \# P \rangle \langle A_P \# Q \rangle \langle A_Q \# Q \rangle$
 $\langle A_P \# R \rangle \langle A_Q \# R \rangle \langle A_P \# M \rangle \langle A_Q \# M \rangle \langle A_P \# A_R \rangle \langle A_Q \# A_R \rangle$
 $\langle A_R \# \Psi \rangle \langle A_R \# P \rangle \langle A_R \# Q \rangle \langle A_R \# R \rangle \langle A_R \# M \rangle \langle xvec \# P \rangle \langle xvec \# R \rangle$
 by (*auto intro: BrComm2*)
 moreover have $(\Psi, ((P' \parallel Q) \parallel R'), (P' \parallel (Q \parallel R'))) \in \text{Rel}$
 by (*rule C1*)
 ultimately show ?*case* by *blast*
 next
 case (*cBrMerge* $\Psi_R Q' A_Q \Psi_Q R' A_R$)
 from $\langle A_{QR} \# N \rangle \langle A_{QR} = A_Q @ A_R \rangle$ have $A_Q \# N$ and $A_R \# N$
 by *simp+*
 from $\langle A_{QR} \# xvec \rangle \langle A_{QR} = (A_Q @ A_R) \rangle$ have $A_Q \# xvec$ and $A_R \# xvec$
 by *simp+*
 have *Aeq*: $A_{QR} = A_Q @ A_R$ and *Ψeq*: $\Psi_{QR} = \Psi_Q \otimes \Psi_R$ by *fact+*
 from *Aeq* $\langle A_{QR} \# P \rangle$ have $A_Q \# P$ and $A_R \# P$ by *simp+*
 from *Aeq* $\langle A_{QR} \# M \rangle$ have $A_Q \# M$ and $A_R \# M$ by *simp+*
 from *Aeq* $\langle A_{QR} \# \Psi \rangle$ have $A_Q \# \Psi$ and $A_R \# \Psi$ by *simp+*
 from *Aeq* $\langle A_P \# A_{QR} \rangle$ have $A_P \# A_Q$ and $A_P \# A_R$ by *simp+*
 with $\langle A_P \# Q \rangle \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ have $A_P \# \Psi_Q$
 by (*metis extractFrameFreshChain freshChainSym freshFrameDest*)
 from *PTrans* *Ψeq* have $(\Psi \otimes \Psi_R) \otimes \Psi_Q \triangleright P \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec P'$
 by (*metis statEqTransition Associativity Commutativity Composition*)
 moreover note *FrP*

moreover from $\langle \Psi \otimes \Psi_P \rangle \otimes \Psi_R \triangleright Q \mapsto_i M(N) \prec Q'$ **have** $\langle \Psi \otimes \Psi_R \rangle$
 $\otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'$
by (*metis statEqTransition Associativity Commutativity Composition*)
moreover note $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
moreover from $\langle A_P \#* Q \rangle \langle A_P \#* R \rangle \langle A_P \#* A_{QR} \rangle \text{Aeq} \langle \text{extractFrame } Q =$
 $\langle A_Q, \Psi_Q \rangle \rangle \langle \text{extractFrame } R = \langle A_R, \Psi_R \rangle \rangle$
have $A_P \#* A_Q$ **and** $A_P \#* \Psi_R$ **by** (*auto dest: extractFrameFreshChain*)
moreover from $\langle A_{QR} \#* \Psi \rangle \langle A_{QR} \#* P \rangle \text{Aeq}$ **have** $A_Q \#* \Psi$ **and** $A_Q \#* P$
by *simp+*
ultimately have $PQTrans: \Psi \otimes \Psi_R \triangleright P \parallel Q \mapsto_i M(\nu * xvec)(N) \prec (P' \parallel$
 $Q')$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_P \#* M \rangle \langle A_P \#* A_Q \rangle \langle A_Q \#* \Psi \rangle$
 $\langle A_Q \#* \Psi_R \rangle \langle A_Q \#* P \rangle \langle A_Q \#* Q \rangle \langle xvec \#* Q \rangle \langle A_Q \#* M \rangle$
by (*intro BrComm2*) (*assumption | force*)
have $RTrans: (\Psi \otimes \Psi_P) \otimes \Psi_Q \triangleright R \mapsto_i M(N) \prec R'$ **and** $FrR: \text{extractFrame}$
 $R = \langle A_R, \Psi_R \rangle$ **by** *fact+*
note $PQTrans$
moreover from $FrP \langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_{QR} \#* \Psi_P \rangle \text{Aeq}$
have $\text{extractFrame}(P \parallel Q) = \langle (A_P @ A_Q), \Psi_P \otimes \Psi_Q \rangle$ **by** *simp+*
moreover from $RTrans$ **have** $\Psi \otimes (\Psi_P \otimes \Psi_Q) \triangleright R \mapsto_i M(N) \prec R'$
by (*metis Associativity statEqTransition*)
moreover note FrR
moreover note $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* Q \rangle \langle A_Q \#* P \rangle \langle A_Q$
 $\#* Q \rangle \langle A_P \#* R \rangle \langle A_Q \#* R \rangle$
 $\langle A_P \#* M \rangle \langle A_Q \#* M \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle \langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R$
 $\#* Q \rangle \langle A_R \#* R \rangle \langle A_R \#* M \rangle \langle xvec \#* P \rangle \langle xvec \#* R \rangle$
ultimately have $\Psi \triangleright (P \parallel Q) \parallel R \mapsto_i M(\nu * xvec)(N) \prec (P' \parallel Q') \parallel R'$
by (*auto intro: BrComm2*)
moreover have $(\Psi, (P' \parallel Q') \parallel R', (P' \parallel (Q' \parallel R'))) \in Rel$
by (*rule C1*)
ultimately show *?case by blast*
qed
qed
qed

lemma *parNilLeft*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$

assumes *eqvt Rel*

and $C1: \bigwedge Q. (\Psi, Q \parallel \mathbf{0}, Q) \in Rel$

shows $\Psi \triangleright (P \parallel \mathbf{0}) \rightsquigarrow_{[Rel]} P$

using *eqvt Rel*

proof (*induct rule: simI[of - - - ()]*)

case (*cSim* $\alpha P'$)

from $\langle \Psi \triangleright P \mapsto_\alpha \prec P' \rangle$ **have** $\Psi \otimes SBottom' \triangleright P \mapsto_\alpha \prec P'$

```

  by(metis statEqTransition Identity Assertion.StatEqSym)
then have  $\Psi \triangleright (P \parallel \mathbf{0}) \mapsto \alpha \prec (P' \parallel \mathbf{0})$ 
  by(rule Par1) auto
moreover have  $(\Psi, P' \parallel \mathbf{0}, P') \in \text{Rel}$  by(rule C1)
ultimately show ?case by blast
qed

lemma parNilRight:
  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c)$  psi
  and  $\text{Rel} :: ('b \times ('a, 'b, 'c)$  psi  $\times ('a, 'b, 'c)$  psi) set

assumes eqvt Rel
  and  $C1: \bigwedge Q. (\Psi, Q, (Q \parallel \mathbf{0})) \in \text{Rel}$ 

shows  $\Psi \triangleright P \rightsquigarrow[\text{Rel}] (P \parallel \mathbf{0})$ 
  using  $\langle \text{eqvt Rel} \rangle$ 
proof(induct rule: simI[of - - - - ()])
  case(cSim  $\alpha$   $P'$ )
  note  $\langle \Psi \triangleright P \parallel \mathbf{0} \mapsto \alpha \prec P' \rangle \langle \text{bn } \alpha \#* \Psi \rangle \langle \text{bn } \alpha \#* P \rangle$ 
  moreover have  $\text{bn } \alpha \#* \mathbf{0}$  by simp
  ultimately show ?case using  $\langle \text{bn } \alpha \#* \text{subject } \alpha \rangle$ 
  proof(induct rule: parCases[where C=()])
    case(cPar1  $P' A_Q \Psi_Q$ )
    from  $\langle \text{extractFrame}(\mathbf{0}) = \langle A_Q, \Psi_Q \rangle \rangle$  have  $\Psi_Q = \text{SBottom}'$  by auto
    with  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle$  have  $\Psi \triangleright P \mapsto \alpha \prec P'$ 
      by(metis statEqTransition Identity)
    moreover have  $(\Psi, P', P' \parallel \mathbf{0}) \in \text{Rel}$  by(rule C1)
    ultimately show ?case by blast
  next
    case(cPar2  $Q' A_P \Psi_P$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto \alpha \prec Q' \rangle$  have False
      by auto
    then show ?case by simp
  next
    case(cComm1  $\Psi_Q M N P' A_P \Psi_P K \text{vec } Q' A_Q$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto K(\nu * \text{vec}) \langle N \rangle \prec Q' \rangle$  have False by auto
    then show ?case by simp
  next
    case(cComm2  $\Psi_Q M \text{vec } N P' A_P \Psi_P K Q' A_Q$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto K \langle N \rangle \prec Q' \rangle$  have False
      by auto
    then show ?case by simp
  next
    case(cBrMerge  $\Psi_Q M N P' A_P \Psi_P Q' A_Q$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto \text{!}M \langle N \rangle \prec Q' \rangle$  have False
      by auto
    then show ?case by simp
  next

```

```

    case(cBrComm1  $\Psi_Q M N P' A_P \Psi_P xvec Q' A_Q$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto jM(\nu*xvec)\langle N \rangle \prec Q' \rangle$  have False
      by auto
    then show ?case by simp
  next
    case(cBrComm2  $\Psi_Q M xvec N P' A_P \Psi_P Q' A_Q$ )
    from  $\langle \Psi \otimes \Psi_P \triangleright \mathbf{0} \mapsto jM\langle N \rangle \prec Q' \rangle$  have False
      by auto
    then show ?case by simp
  qed
qed

```

lemma *resNilLeft*:

```

  fixes  $x :: name$ 
  and  $\Psi :: 'b$ 
  and  $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

```

```

shows  $\Psi \triangleright (\nu x)\mathbf{0} \rightsquigarrow[Rel] \mathbf{0}$ 
by(auto simp add: simulation-def)

```

lemma *resNilRight*:

```

  fixes  $x :: name$ 
  and  $\Psi :: 'b$ 
  and  $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$ 

```

```

shows  $\Psi \triangleright \mathbf{0} \rightsquigarrow[Rel] (\nu x)\mathbf{0}$ 
apply(clarsimp simp add: simulation-def)
by(cases rule: semantics.cases) (auto simp add: psi.inject alpha')

```

lemma *inputPushResLeft*:

```

  fixes  $\Psi :: 'b$ 
  and  $x :: name$ 
  and  $M :: 'a$ 
  and  $xvec :: name list$ 
  and  $N :: 'a$ 
  and  $P :: ('a, 'b, 'c) psi$ 

```

assumes *eqvt Rel*

```

  and  $x \# \Psi$ 
  and  $x \# M$ 
  and  $x \# xvec$ 
  and  $x \# N$ 
  and  $C1: \bigwedge Q. (\Psi, Q, Q) \in Rel$ 

```

```

shows  $\Psi \triangleright (\nu x)(M(\lambda*xvec N).P) \rightsquigarrow[Rel] M(\lambda*xvec N).(\nu x)P$ 

```

proof –

```

  note  $\langle eqvt Rel \rangle \langle x \# \Psi \rangle$ 
  moreover have  $x \# (\nu x)(M(\lambda*xvec N).P)$  by(simp add: abs-fresh)
  moreover from  $\langle x \# M \rangle \langle x \# N \rangle$  have  $x \# M(\lambda*xvec N).(\nu x)P$ 

```

```

    by(auto simp add: inputChainFresh abs-fresh)
  ultimately show ?thesis
proof(induct rule: simIFresh[of - - - - ()])
  case(cSim  $\alpha$   $P'$ )
  from  $\langle \Psi \triangleright M(\lambda * xvec\ N).(\nu x)P \mapsto \alpha \prec P' \rangle \langle x \# \alpha \rangle$  show ?case
  proof(induct rule: inputCases)
    case(cInput  $K$   $Tvec$ )
    from  $\langle x \# K(N[xvec::=Tvec]) \rangle$  have  $x \# K$  and  $x \# N[xvec::=Tvec]$  by simp+
    from  $\langle \Psi \vdash M \leftrightarrow K \rangle \langle distinct\ xvec \rangle \langle set\ xvec \subseteq supp\ N \rangle \langle length\ xvec = length\ Tvec \rangle$ 
    have  $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto K(N[xvec::=Tvec]) \prec P[xvec::=Tvec]$ 
    by(rule Input)
    then have  $\Psi \triangleright (\nu x)(M(\lambda * xvec\ N).P) \mapsto K(N[xvec::=Tvec]) \prec (\nu x)(P[xvec::=Tvec])$ 
  using  $\langle x \# \Psi \rangle \langle x \# K \rangle \langle x \# N[xvec::=Tvec] \rangle$ 
    by(intro Scope) auto
    moreover from  $\langle length\ xvec = length\ Tvec \rangle \langle distinct\ xvec \rangle \langle set\ xvec \subseteq supp\ N \rangle \langle x \# N[xvec::=Tvec] \rangle$  have  $x \# Tvec$ 
    by(rule substTerm.subst3)
    with  $\langle x \# xvec \rangle$  have  $(\Psi, (\nu x)(P[xvec::=Tvec]), ((\nu x)P)[xvec::=Tvec]) \in Rel$ 
    by(force intro: C1)
    ultimately show ?case by blast
  next
  case(cBrInput  $K$   $Tvec$ )
  from  $\langle x \# \iota K(N[xvec::=Tvec]) \rangle$  have  $x \# K$  and  $x \# N[xvec::=Tvec]$  by simp+
  from  $\langle \Psi \vdash K \succeq M \rangle \langle distinct\ xvec \rangle \langle set\ xvec \subseteq supp\ N \rangle \langle length\ xvec = length\ Tvec \rangle$ 
  have  $\Psi \triangleright M(\lambda * xvec\ N).P \mapsto \iota K(N[xvec::=Tvec]) \prec P[xvec::=Tvec]$ 
  by(rule BrInput)
  then have  $\Psi \triangleright (\nu x)(M(\lambda * xvec\ N).P) \mapsto \iota K(N[xvec::=Tvec]) \prec (\nu x)(P[xvec::=Tvec])$ 
  using  $\langle x \# \Psi \rangle \langle x \# K \rangle \langle x \# N[xvec::=Tvec] \rangle$ 
  by(intro Scope) auto
  moreover from  $\langle length\ xvec = length\ Tvec \rangle \langle distinct\ xvec \rangle \langle set\ xvec \subseteq supp\ N \rangle \langle x \# N[xvec::=Tvec] \rangle$  have  $x \# Tvec$ 
  by(rule substTerm.subst3)
  with  $\langle x \# xvec \rangle$  have  $(\Psi, (\nu x)(P[xvec::=Tvec]), ((\nu x)P)[xvec::=Tvec]) \in Rel$ 
  by(force intro: C1)
  ultimately show ?case by blast
qed
qed
qed

```

lemma *inputPushResRight*:

```

fixes  $\Psi$    :: 'b
and  $x$      :: name
and  $M$     :: 'a
and  $xvec$  :: name list
and  $N$     :: 'a
and  $P$     :: ('a, 'b, 'c) psi

```



```

assumes eqvt Rel
and  $x \# \Psi$ 
and  $x \# M$ 
and  $x \# \mathit{xvec}$ 
and  $x \# N$ 
and  $C1: \bigwedge Q. (\Psi, Q, Q) \in \mathit{Rel}$ 

shows  $\Psi \triangleright M(\lambda * \mathit{xvec} N).(\nu x)P \rightsquigarrow[\mathit{Rel}] (\nu x)(M(\lambda * \mathit{xvec} N).P)$ 
proof –
  note  $\langle \mathit{eqvt Rel} \rangle \langle x \# \Psi \rangle$ 
  moreover from  $\langle x \# M \rangle \langle x \# N \rangle$  have  $x \# M(\lambda * \mathit{xvec} N).(\nu x)P$ 
    by (auto simp add: inputChainFresh abs-fresh)
  moreover have  $x \# (\nu x)(M(\lambda * \mathit{xvec} N).P)$  by (simp add: abs-fresh)
  ultimately show ?thesis
  proof (induct rule: simIFresh[of - - - - ()])
    case (cSim  $\alpha P'$ )
      note  $\langle \Psi \triangleright (\nu x)(M(\lambda * \mathit{xvec} N).P) \mapsto \alpha \prec P' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# P' \rangle \langle \mathit{bn} \alpha \# * \Psi \rangle$ 
      moreover from  $\langle \mathit{bn} \alpha \# * ((\nu x)(M(\lambda * \mathit{xvec} N).P)) \rangle \langle x \# \alpha \rangle$  have  $\mathit{bn} \alpha \# * (M(\lambda * \mathit{xvec} N).P)$ 
        by simp
      ultimately show ?case using  $\langle \mathit{bn} \alpha \# * \mathit{subject} \alpha \rangle$ 
      proof (induct rule: resCases[where C=()])
        case (cRes  $P'$ )
          from  $\langle \Psi \triangleright M(\lambda * \mathit{xvec} N).P \mapsto \alpha \prec P' \rangle \langle x \# \alpha \rangle$  show ?case
          proof (induct rule: inputCases)
            case (cInput  $K Tvec$ )
              from  $\langle x \# K(N[\mathit{xvec} ::= Tvec]) \rangle$  have  $x \# K$  and  $x \# N[\mathit{xvec} ::= Tvec]$  by simp+
              from  $\langle \Psi \vdash M \leftrightarrow K \rangle \langle \mathit{distinct} \mathit{xvec} \rangle \langle \mathit{set} \mathit{xvec} \subseteq \mathit{supp} N \rangle \langle \mathit{length} \mathit{xvec} = \mathit{length} Tvec \rangle$ 
                have  $\Psi \triangleright M(\lambda * \mathit{xvec} N).(\nu x)P \mapsto K((N[\mathit{xvec} ::= Tvec])) \prec ((\nu x)P)[\mathit{xvec} ::= Tvec]$ 
                  by (rule Input)
                moreover from  $\langle \mathit{length} \mathit{xvec} = \mathit{length} Tvec \rangle \langle \mathit{distinct} \mathit{xvec} \rangle \langle \mathit{set} \mathit{xvec} \subseteq \mathit{supp} N \rangle \langle x \# N[\mathit{xvec} ::= Tvec] \rangle$  have  $x \# Tvec$ 
                  by (rule substTerm.subst3)
                with  $\langle x \# \mathit{xvec} \rangle$  have  $(\Psi, ((\nu x)P)[\mathit{xvec} ::= Tvec], (\nu x)(P[\mathit{xvec} ::= Tvec])) \in \mathit{Rel}$ 
                  by (force intro: C1)
                ultimately show ?case by blast
              next
                case (cBrInput  $K Tvec$ )
                  from  $\langle x \# \mathit{i}K((N[\mathit{xvec} ::= Tvec])) \rangle$  have  $x \# K$  and  $x \# N[\mathit{xvec} ::= Tvec]$  by simp+
                  from  $\langle \Psi \vdash K \succeq M \rangle \langle \mathit{distinct} \mathit{xvec} \rangle \langle \mathit{set} \mathit{xvec} \subseteq \mathit{supp} N \rangle \langle \mathit{length} \mathit{xvec} = \mathit{length} Tvec \rangle$ 
                    have  $\Psi \triangleright M(\lambda * \mathit{xvec} N).(\nu x)P \mapsto \mathit{i}K((N[\mathit{xvec} ::= Tvec])) \prec ((\nu x)P)[\mathit{xvec} ::= Tvec]$ 
                      by (rule BrInput)
                    moreover from  $\langle \mathit{length} \mathit{xvec} = \mathit{length} Tvec \rangle \langle \mathit{distinct} \mathit{xvec} \rangle \langle \mathit{set} \mathit{xvec} \subseteq \mathit{supp} N \rangle$ 

```

```

N⟩ ⟨x # N[xvec::=Tvec]⟩ have x # Tvec
  by(rule substTerm.subst3)
with ⟨x # xvec⟩ have (Ψ, ((νx)P)[xvec::=Tvec], (νx)(P[xvec::=Tvec])) ∈
Rel
  by(force intro: C1)
  ultimately show ?case by blast
qed
next
  case cOpen
  then have False by auto
  then show ?case by simp
next
  case cBrOpen
  then have False by auto
  then show ?case by simp
next
  case (cBrClose M xvec N P′)
  then have False by auto
  then show ?case by simp
qed
qed
qed

```

```

lemma outputPushResLeft:
fixes Ψ :: 'b
  and x :: name
  and M :: 'a
  and N :: 'a
  and P :: ('a, 'b, 'c) psi

```

```

assumes eqvt Rel
and x # Ψ
and x # M
and x # N
and C1: ∧Q. (Ψ, Q, Q) ∈ Rel

```

```

shows Ψ ▷ (νx)(M⟨N⟩.P) ~>[Rel] M⟨N⟩.(νx)P

```

```

proof -
note ⟨eqvt Rel⟩ ⟨x # Ψ⟩
moreover have x # (νx)(M⟨N⟩.P) by(simp add: abs-fresh)
moreover from ⟨x # M⟩ ⟨x # N⟩ have x # M⟨N⟩.(νx)P
  by(auto simp add: abs-fresh)
ultimately show ?thesis
proof(induct rule: simIFresh[of - - - - ()])
  case(cSim α P′)
  from ⟨Ψ ▷ M⟨N⟩.(νx)P ⟶α < P′⟩ ⟨x # α⟩
  show ?case
proof(induct rule: outputCases)
  case(cOutput K)

```

```

from  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \triangleright M\langle N \rangle.P \mapsto K\langle N \rangle \prec P$ 
  by(rule Output)
then have  $\Psi \triangleright (\nu x)(M\langle N \rangle.P) \mapsto K\langle N \rangle \prec (\nu x)P$  using  $\langle x \# \Psi \rangle \langle x \# K\langle N \rangle \rangle$ 
  by(rule Scope)
moreover have  $(\Psi, (\nu x)P, (\nu x)P) \in Rel$  by(rule C1)
ultimately show ?case by blast
next
case(cBrOutput K)
from  $\langle \Psi \vdash M \preceq K \rangle$  have  $\Psi \triangleright M\langle N \rangle.P \mapsto_i K\langle N \rangle \prec P$ 
  by(rule BrOutput)
then have  $\Psi \triangleright (\nu x)(M\langle N \rangle.P) \mapsto_i K\langle N \rangle \prec (\nu x)P$  using  $\langle x \# \Psi \rangle \langle x \#$ 
 $_i K\langle N \rangle \rangle$ 
  by(rule Scope)
moreover have  $(\Psi, (\nu x)P, (\nu x)P) \in Rel$  by(rule C1)
ultimately show ?case by blast
qed
qed
qed

```

lemma *broutputNoBind*:

```

fixes  $\Psi :: 'b$ 
  and  $M :: 'a$ 
  and  $N :: 'a$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $\alpha :: 'a action$ 
  and  $P' :: ('a, 'b, 'c) psi$ 

```

assumes $\Psi \triangleright M\langle N \rangle.P \mapsto ({}_i K(\nu * xvec)\langle N' \rangle) \prec P'$

shows $xvec = []$

proof –

from *assms* **have** $bn({}_i K(\nu * xvec)\langle N' \rangle) = []$

by(*induct rule: outputCases*) *auto*

then show ?thesis **by** *simp*

qed

lemma *broutputObjectEq*:

```

fixes  $\Psi :: 'b$ 
  and  $M :: 'a$ 
  and  $N :: 'a$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $\alpha :: 'a action$ 
  and  $P' :: ('a, 'b, 'c) psi$ 

```

assumes $\Psi \triangleright M\langle N \rangle.P \mapsto ({}_i K(\nu * xvec)\langle N' \rangle) \prec P'$

shows $N = N'$

proof –

from *assms* **have** $object({}_i K(\nu * xvec)\langle N' \rangle) = Some N$

by(*induct rule: outputCases*) *auto*

then show ?thesis

by *simp*
qed

lemma *brOutputOutputCases*[*consumes 1, case-names cBrOutput*]:

fixes $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c) \text{ psi}$
and $\alpha :: 'a \text{ action}$
and $P' :: ('a, 'b, 'c) \text{ psi}$

assumes *Trans*: $\Psi \triangleright M \langle N \rangle . P \mapsto (\text{jK}(\nu * xvec) \langle N' \rangle) \prec P'$
and *rBrOutput*: $\bigwedge K. \Psi \vdash M \preceq K \implies \text{Prop} (\text{jK} \langle N \rangle) P$

shows $\text{Prop} (\text{jK}(\nu * xvec) \langle N' \rangle) P'$

proof –

have $xvec = []$ **using** *Trans* **by**(*rule broutputNoBind*)
then obtain $K' N''$ **where** *eq*: $(\text{jK}(\nu * xvec) \langle N' \rangle) = \text{jK}' \langle N'' \rangle$
by *blast*
have $N = N'$ **using** *Trans* **by**(*rule broutputObjectEq*)
from *Trans* $\langle (\text{jK}(\nu * xvec) \langle N' \rangle) = \text{jK}' \langle N'' \rangle \rangle$
show *?thesis* **unfolding** $\langle N = N' \rangle$ [*symmetric*]
proof(*induct rule: outputCases*)
case (*cOutput K*) then show *?case* **by** *simp*
next
case (*cBrOutput K*) then show *?case*
by(*intro rBrOutput*)

qed

qed

lemma *outputPushResRight*:

fixes $\Psi :: 'b$
and $x :: \text{name}$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c) \text{ psi}$

assumes *eqvt Rel*

and $x \# \Psi$
and $x \# M$
and $x \# N$
and *C1*: $\bigwedge Q. (\Psi, Q, Q) \in \text{Rel}$

shows $\Psi \triangleright M \langle N \rangle . (\nu x) P \rightsquigarrow [\text{Rel}] (\nu x) (M \langle N \rangle . P)$

proof –

note $\langle \text{eqvt Rel} \rangle \langle x \# \Psi \rangle$
moreover from $\langle x \# M \rangle \langle x \# N \rangle$ **have** $x \# M \langle N \rangle . (\nu x) P$
by(*auto simp add: abs-fresh*)
moreover have $x \# (\nu x) (M \langle N \rangle . P)$ by(*simp add: abs-fresh*)

```

ultimately show ?thesis
proof(induct rule: simIFresh[of - - - - (M, N)])
  case(cSim  $\alpha$   $P'$ )
  note  $\langle \Psi \triangleright (\nu x)(M\langle N \rangle.P) \mapsto \alpha \prec P' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# P' \rangle \langle bn \alpha \# \Psi \rangle$ 
  moreover from  $\langle bn \alpha \# (\nu x)(M\langle N \rangle.P) \rangle \langle x \# \alpha \rangle$  have  $bn \alpha \# (M\langle N \rangle.P)$ 
by simp
  ultimately show ?case using  $\langle bn \alpha \# \text{subject } \alpha \rangle \langle bn \alpha \# (M, N) \rangle \langle x \# \alpha \rangle$ 
   $\langle x \# M \rangle \langle x \# N \rangle$ 
  proof(induct rule: resCases[where  $C=()$ ])
  case(cOpen  $K$   $xvec1$   $xvec2$   $y$   $N' P'$ )
  from  $\langle bn(K(\nu*(xvec1@y\#xvec2))\langle N' \rangle) \# (M, N) \rangle$  have  $y \# N$  by simp+
  from  $\langle \Psi \triangleright M\langle N \rangle.P \mapsto K(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot P' \rangle$ 
  have  $N = [(x, y)] \cdot N'$ 
  apply -
  by(ind-cases  $\Psi \triangleright M\langle N \rangle.P \mapsto K(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot P'$ )
  (auto simp add: residualInject psi.inject)
  with  $\langle x \# N \rangle \langle y \# N \rangle \langle x \neq y \rangle$  have  $N = N'$ 
  by(subst pt-bij[OF pt-name-inst, OF at-name-inst, symmetric, where  $pi=[(x, y)]$ ])
  (simp add: fresh-left calc-atm)
  with  $\langle y \in \text{supp } N' \rangle \langle y \# N \rangle$  have False by(simp add: fresh-def)
  then show ?case by simp
next
  case(cBrOpen  $K$   $xvec1$   $xvec2$   $y$   $N' P'$ )
  from  $\langle bn(\downarrow K(\nu*(xvec1@y\#xvec2))\langle N' \rangle) \# (M, N) \rangle$  have  $y \# N$  by simp+
  from  $\langle \Psi \triangleright M\langle N \rangle.P \mapsto \downarrow K(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot P' \rangle$ 
  have  $N = [(x, y)] \cdot N'$ 
  apply -
  by(ind-cases  $\Psi \triangleright M\langle N \rangle.P \mapsto \downarrow K(\nu*(xvec1@xvec2))\langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot P'$ )
  (auto simp add: residualInject psi.inject)
  with  $\langle x \# N \rangle \langle y \# N \rangle \langle x \neq y \rangle$  have  $N = N'$ 
  by(subst pt-bij[OF pt-name-inst, OF at-name-inst, symmetric, where  $pi=[(x, y)]$ ])
  (simp add: fresh-left calc-atm)
  with  $\langle y \in \text{supp } N' \rangle \langle y \# N \rangle$  have False by(simp add: fresh-def)
  then show ?case by simp
next
  case(cRes  $P'$ )
  from  $\langle \Psi \triangleright M\langle N \rangle.P \mapsto \alpha \prec P' \rangle$  show ?case
proof(induct rule: outputCases)
  case(cOutput  $K$ )
  from  $\langle \Psi \vdash M \leftrightarrow K \rangle$  have  $\Psi \triangleright M\langle N \rangle.((\nu x)P) \mapsto K\langle N \rangle \prec (\nu x)P$ 
  by(rule Output)
  moreover have  $(\Psi, (\nu x)P, (\nu x)P) \in \text{Rel}$  by(rule C1)
  ultimately show ?case by force

```

```

next
  case(cBrOutput K)
  from ⟨Ψ ⊢ M ≲ K⟩ have Ψ ▷ M⟨N⟩.⟨(νx)P⟩ ⟶iK⟨N⟩ ≲ (νx)P
  by(rule BrOutput)
  moreover have (Ψ, (νx)P, (νx)P) ∈ Rel by(rule C1)
  ultimately show ?case by force
qed
next
case(cBrClose K xvec N' P')
from ⟨Ψ ▷ M⟨N⟩.P ⟶iK⟨ν*xvec⟩⟨N'⟩ ≲ P'⟩
have Ψ ⊢ M ≲ the(subject(iK⟨ν*xvec⟩⟨N'⟩))
  by(rule brOutputOutputCases) simp
then have Ψ ⊢ M ≲ K by simp
then have supp K ⊆ (supp M :: name set) by(rule chanOutConSupp)
then have False using ⟨x ∈ supp K⟩ ⟨x # M⟩ unfolding fresh-def
  by blast
then show ?case by(rule FalseE)
qed
qed
qed
lemma casePushResLeft:
  fixes Ψ :: 'b
  and x :: name
  and Cs :: ('c × ('a, 'b, 'c) psi) list

assumes eqvt Rel
  and x # Ψ
  and x # map fst Cs
  and C1: ⋀Q. (Ψ, Q, Q) ∈ Rel

shows Ψ ▷ (νx)(Cases Cs) ∼[Rel] Cases (map (λ(φ, P). (φ, (νx)P)) Cs)
proof -
  note ⟨eqvt Rel⟩ ⟨x # Ψ⟩
  moreover have x # (νx)(Cases Cs) by(simp add: abs-fresh)
  moreover from ⟨x # map fst Cs⟩ have x # Cases (map (λ(φ, P). (φ, (νx)P)) Cs)
  by(induct Cs) (auto simp add: abs-fresh)
  ultimately show ?thesis
  proof(induct rule: simIFresh[of - - - - Cs])
    case(cSim α P'')
    from ⟨Ψ ▷ Cases (map (λ(φ, P). (φ, (νx)P)) Cs) ⟶iα ≲ P''⟩
    show ?case
  proof(induct rule: caseCases)
    case(cCase φ P')
    from ⟨(φ, P') ∈ set (map (λ(φ, P). (φ, (νx)P)) Cs)⟩
    obtain P where (φ, P) ∈ set Cs and P' = (νx)P
    by(induct Cs) auto
    from ⟨guarded P'⟩ ⟨P' = (νx)P⟩ have guarded P by simp
  end
end
end

```

```

from  $\langle \Psi \triangleright P' \mapsto \alpha \prec P'' \rangle \langle P' = (\nu x)P \rangle$  have  $\Psi \triangleright (\nu x)P \mapsto \alpha \prec P''$ 
  by simp
moreover note  $\langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# P'' \rangle \langle \text{bn } \alpha \# \Psi \rangle$ 
moreover from  $\langle \text{bn } \alpha \# Cs \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$ 
have  $\text{bn } \alpha \# P$  by (auto dest: memFreshChain)
ultimately show ?case using  $\langle \text{bn } \alpha \# \text{subject } \alpha \rangle \langle x \# \alpha \rangle \langle \text{bn } \alpha \# Cs \rangle$ 
proof (induct rule: resCases[where C=())
  case (cOpen M xvec1 xvec2 y N P')
    from  $\langle x \# M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$  have  $x \# xvec1$  and  $x \# xvec2$  and
 $x \# M$  by simp+
    from  $\langle \text{bn}(M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle) \# Cs \rangle$  have  $y \# Cs$  by simp
    from  $\langle \Psi \triangleright P \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle ((x, y) \cdot N) \rangle \prec \langle ((x, y) \cdot P') \rangle \langle (\varphi,$ 
 $P) \in \text{set } Cs \rangle \langle \Psi \vdash \varphi \rangle \langle \text{guarded } P \rangle$ 
    have  $\Psi \triangleright \text{Cases } Cs \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle ((x, y) \cdot N) \rangle \prec \langle ((x, y) \cdot$ 
 $P') \rangle$  by (rule Case)
    then have  $\langle ((x, y) \cdot \Psi) \rangle \triangleright \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto \langle ((x, y) \cdot (M(\nu*(xvec1@xvec2)) \rangle \langle ((x,$ 
 $y) \cdot N) \rangle \prec \langle ((x, y) \cdot P') \rangle)$ 
    by (rule semantics.eqt)
    with  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle$ 
 $\langle x \# xvec2 \rangle$ 
    have  $\Psi \triangleright \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle N \rangle \prec P'$  by (simp
add: eqts)
    then have  $\Psi \triangleright (\nu y) \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$ 
 $\prec P'$  using  $\langle y \in \text{supp } N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ 
    by (rule Open)
    then have  $\Psi \triangleright (\nu x) \langle \text{Cases } Cs \rangle \mapsto M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle \prec P'$ 
using  $\langle y \# Cs \rangle$ 
    by (simp add: alphaRes)
    moreover have  $(\Psi, P', P') \in \text{Rel}$  by (rule C1)
    ultimately show ?case by blast
  next
  case (cBrOpen M xvec1 xvec2 y N P')
    from  $\langle x \# M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$  have  $x \# xvec1$  and  $x \# xvec2$  and
 $x \# M$  by simp+
    from  $\langle \text{bn}(M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle) \# Cs \rangle$  have  $y \# Cs$  by simp
    from  $\langle \Psi \triangleright P \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle ((x, y) \cdot N) \rangle \prec \langle ((x, y) \cdot P') \rangle$ 
 $\langle (\varphi, P) \in \text{set } Cs \rangle \langle \Psi \vdash \varphi \rangle \langle \text{guarded } P \rangle$ 
    have  $\Psi \triangleright \text{Cases } Cs \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle ((x, y) \cdot N) \rangle \prec \langle ((x, y) \cdot$ 
 $P') \rangle$  by (rule Case)
    then have  $\langle ((x, y) \cdot \Psi) \rangle \triangleright \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto \langle ((x, y) \cdot (M(\nu*(xvec1@xvec2)) \rangle \langle ((x,$ 
 $y) \cdot N) \rangle \prec \langle ((x, y) \cdot P') \rangle)$ 
    by (rule semantics.eqt)
    with  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle$ 
 $\langle x \# xvec2 \rangle$ 
    have  $\Psi \triangleright \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto M(\nu*(xvec1@xvec2)) \rangle \langle N \rangle \prec P'$ 
by (simp add: eqts)
    then have  $\Psi \triangleright (\nu y) \langle ((x, y) \cdot (\text{Cases } Cs)) \rangle \mapsto M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$ 
 $\prec P'$  using  $\langle y \in \text{supp } N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ 
    by (rule BrOpen)

```

```

    then have  $\Psi \triangleright (\nu x)(Cases\ Cs) \mapsto_{iM}(\nu*(xvec1@y\#xvec2))\langle N \rangle \prec P'$ 
using  $\langle y \# Cs \rangle$ 
  by(simp add: alphaRes)
  moreover have  $(\Psi, P', P') \in Rel$  by(rule C1)
  ultimately show ?case by blast
next
case(cRes P')
from  $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle (\varphi, P) \in set\ Cs \rangle \langle \Psi \vdash \varphi \rangle \langle guarded\ P \rangle$ 
have  $\Psi \triangleright Cases\ Cs \mapsto \alpha \prec P'$  by(rule Case)
then have  $\Psi \triangleright (\nu x)(Cases\ Cs) \mapsto \alpha \prec (\nu x)P'$  using  $\langle x \# \Psi \rangle \langle x \# \alpha \rangle$ 
  by(rule Scope)
  moreover have  $(\Psi, (\nu x)P', (\nu x)P') \in Rel$  by(rule C1)
  ultimately show ?case by blast
next
case(cBrClose M xvec N P')
from  $\langle \Psi \triangleright P \mapsto_{iM}(\nu*xvec)\langle N \rangle \prec P' \rangle \langle x \in supp\ M \rangle \langle x \# \Psi \rangle$ 
have  $\Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)((\nu*xvec)P')$ 
  by(rule BrClose)
from  $\langle \Psi \triangleright P \mapsto_{iM}(\nu*xvec)\langle N \rangle \prec P' \rangle \langle (\varphi, P) \in set\ Cs \rangle \langle \Psi \vdash \varphi \rangle \langle guarded\ P \rangle$ 
  have  $\Psi \triangleright Cases\ Cs \mapsto_{iM}(\nu*xvec)\langle N \rangle \prec P'$ 
  by(rule Case)
  then have  $\Psi \triangleright (\nu x)(Cases\ Cs) \mapsto \tau \prec (\nu x)((\nu*xvec)P')$  using  $\langle x \in supp\ M \rangle \langle x \# \Psi \rangle$ 
  by(rule BrClose)
  moreover have  $(\Psi, (\nu x)((\nu*xvec)P'), (\nu x)((\nu*xvec)P')) \in Rel$  by fact
  ultimately show ?case by blast
qed
qed
qed
qed

```

lemma casePushResRight:

```

fixes  $\Psi :: 'b$ 
and  $x :: name$ 
and  $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$ 

```

assumes eqvt Rel

```

and  $x \# \Psi$ 
and  $x \# map\ fst\ Cs$ 
and  $C1: \bigwedge Q. (\Psi, Q, Q) \in Rel$ 

```

shows $\Psi \triangleright Cases\ (map\ (\lambda(\varphi, P). (\varphi, (\nu x)P))\ Cs) \rightsquigarrow_{[Rel]} (\nu x)(Cases\ Cs)$

proof –

```

note  $\langle eqvt\ Rel \rangle \langle x \# \Psi \rangle$ 

```

```

moreover from  $\langle x \# map\ fst\ Cs \rangle$  have  $x \# Cases\ (map\ (\lambda(\varphi, P). (\varphi, (\nu x)P))\ Cs)$ 

```

```

by(induct Cs) (auto simp add: abs-fresh)

```

```

moreover have  $x \# (\nu x)(Cases\ Cs)$  by(simp add: abs-fresh)

```



```

ultimately show ?thesis
proof(induct rule: simIFresh[of - - - - Cs])
  case(cSim  $\alpha$   $P''$ )
  note  $\langle \Psi \triangleright (\nu x)(\nu y)(Cases\ Cs) \mapsto \alpha \prec P'' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# P'' \rangle \langle bn\ \alpha \# \Psi \rangle$ 
  moreover from  $\langle bn\ \alpha \# (\nu x)(\nu y)(Cases\ Cs) \rangle \langle x \# \alpha \rangle$  have  $bn\ \alpha \# (Cases\ Cs)$ 
by simp
ultimately show ?case using  $\langle bn\ \alpha \# subject\ \alpha \rangle \langle x \# \alpha \rangle \langle bn\ \alpha \# Cs \rangle$ 
proof(induct rule: resCases[where  $C=()$ ])
  case(cOpen  $M\ xvec1\ xvec2\ y\ N\ P'$ )
  from  $\langle x \# M(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$  have  $x \# xvec1$  and  $x \# xvec2$  and
 $x \# M$  by simp+
  from  $\langle bn(M(\nu*(xvec1@y\#xvec2)) \langle N \rangle) \# Cs \rangle$  have  $y \# Cs$  by simp
  from  $\langle \Psi \triangleright Cases\ Cs \mapsto M(\nu*(xvec1@xvec2)) \langle ((x, y) \cdot N) \rangle \prec ((x, y) \cdot P') \rangle$ 
  show ?case
  proof(induct rule: caseCases)
    case(cCase  $\varphi\ P$ )
    from  $\langle \Psi \triangleright P \mapsto M(\nu*(xvec1@xvec2)) \langle ((x, y) \cdot N) \rangle \prec ((x, y) \cdot P') \rangle$ 
    have  $\langle ((x, y) \cdot \Psi) \triangleright ((x, y) \cdot P) \mapsto ((x, y) \cdot (M(\nu*(xvec1@xvec2)) \langle ((x, y) \cdot N) \rangle \prec ((x, y) \cdot P')))) \rangle$ 
    by(rule semantics.eqt)
    with  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle$ 
 $\langle x \# xvec2 \rangle$ 
    have  $\Psi \triangleright ((x, y) \cdot P) \mapsto M(\nu*(xvec1@xvec2)) \langle N \rangle \prec P'$  by(simp add: eqts)
    then have  $\Psi \triangleright (\nu y)((x, y) \cdot P) \mapsto M(\nu*(xvec1@y\#xvec2)) \langle N \rangle \prec P'$ 
  using  $\langle y \in supp\ N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ 
  by(rule Open)
  then have  $\Psi \triangleright (\nu x)P \mapsto M(\nu*(xvec1@y\#xvec2)) \langle N \rangle \prec P'$  using  $\langle y \# Cs \rangle \langle (\varphi, P) \in set\ Cs \rangle$ 
  by(subst alphaRes, auto dest: memFresh)
  moreover from  $\langle (\varphi, P) \in set\ Cs \rangle$  have  $(\varphi, (\nu x)P) \in set\ (map\ (\lambda(\varphi, P). (\varphi, (\nu x)P)))\ Cs$ 
  by(induct Cs) auto
  moreover note  $\langle \Psi \vdash \varphi \rangle$ 
  moreover from  $\langle guarded\ P \rangle$  have  $guarded((\nu x)P)$  by simp
  ultimately have  $\Psi \triangleright (Cases\ (map\ (\lambda(\varphi, P). (\varphi, (\nu x)P)))\ Cs) \mapsto M(\nu*(xvec1@y\#xvec2)) \langle N \rangle$ 
 $\prec P'$ 
  by(rule Case)
  moreover have  $(\Psi, P', P') \in Rel$  by(rule C1)
  ultimately show ?case by blast
qed
next
case(cBrOpen  $M\ xvec1\ xvec2\ y\ N\ P'$ )
  from  $\langle x \# iM(\nu*(xvec1@y\#xvec2)) \rangle \langle N \rangle$  have  $x \# xvec1$  and  $x \# xvec2$  and
 $x \# M$  by simp+
  from  $\langle bn(iM(\nu*(xvec1@y\#xvec2)) \langle N \rangle) \# Cs \rangle$  have  $y \# Cs$  by simp
  from  $\langle \Psi \triangleright Cases\ Cs \mapsto iM(\nu*(xvec1@xvec2)) \langle ((x, y) \cdot N) \rangle \prec ((x, y) \cdot P') \rangle$ 

```

```

show ?case
proof(induct rule: caseCases)
  case(cCase  $\varphi$   $P$ )
    from  $\langle \Psi \triangleright P \mapsto_{iM} (\nu^*(xvec1 @ xvec2)) \langle [(x, y)] \cdot N \rangle \prec [(x, y)] \cdot P' \rangle$ 
    have  $\langle [(x, y)] \cdot \Psi \triangleright [(x, y)] \cdot P \mapsto \langle [(x, y)] \cdot (iM (\nu^*(xvec1 @ xvec2)) \langle [(x, y)] \cdot N \rangle \prec [(x, y)] \cdot P') \rangle$ 
    by(rule semantics.eqvt)
    with  $\langle x \# \Psi \rangle \langle x \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle x \# xvec1 \rangle$ 
     $\langle x \# xvec2 \rangle$ 
    have  $\Psi \triangleright \langle [(x, y)] \cdot P \mapsto_{iM} (\nu^*(xvec1 @ xvec2)) \langle N \rangle \prec P' \mathbf{by}$ (simp add:
    eqts)
    then have  $\Psi \triangleright \langle (\nu y) \langle [(x, y)] \cdot P \mapsto_{iM} (\nu^*(xvec1 @ y \# xvec2)) \langle N \rangle \prec P' \rangle$ 
using  $\langle y \in \text{supp } N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ 
    by(rule BrOpen)
    then have  $\Psi \triangleright \langle (\nu x) P \mapsto_{iM} (\nu^*(xvec1 @ y \# xvec2)) \langle N \rangle \prec P' \mathbf{using}$   $\langle y \#$ 
     $Cs \rangle \langle (\varphi, P) \in \text{set } Cs \rangle$ 
    by(subst alphaRes, auto dest: memFresh)
    moreover from  $\langle (\varphi, P) \in \text{set } Cs \rangle \mathbf{have}$   $\langle \varphi, (\nu x) P \rangle \in \text{set } (\text{map } (\lambda(\varphi, P).$ 
     $(\varphi, (\nu x) P)) Cs)$ 
    by(induct Cs auto)
    moreover note  $\langle \Psi \vdash \varphi \rangle$ 
    moreover from  $\langle \text{guarded } P \rangle \mathbf{have}$   $\langle \text{guarded } ((\nu x) P) \mathbf{by}$  simp
    ultimately have  $\Psi \triangleright \langle \text{Cases } (\text{map } (\lambda(\varphi, P). (\varphi, (\nu x) P)) Cs) \mapsto_{iM} (\nu^*(xvec1 @ y \# xvec2)) \langle N \rangle$ 
     $\prec P' \mathbf{by}$ (rule Case)
    moreover have  $\langle \Psi, P', P' \rangle \in \text{Rel} \mathbf{by}$ (rule C1)
    ultimately show ?case by blast
qed
next
  case(cRes  $P'$ )
  from  $\langle \Psi \triangleright \text{Cases } Cs \mapsto \alpha \prec P' \rangle$ 
  show ?case
  proof(induct rule: caseCases)
    case(cCase  $\varphi$   $P$ )
      from  $\langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle$ 
      have  $\Psi \triangleright \langle (\nu x) P \mapsto \alpha \prec (\nu x) P' \mathbf{by}$ (rule Scope)
      moreover from  $\langle (\varphi, P) \in \text{set } Cs \rangle \mathbf{have}$   $\langle \varphi, (\nu x) P \rangle \in \text{set } (\text{map } (\lambda(\varphi, P).$ 
       $(\varphi, (\nu x) P)) Cs)$ 
      by(induct Cs auto)
      moreover note  $\langle \Psi \vdash \varphi \rangle$ 
      moreover from  $\langle \text{guarded } P \rangle \mathbf{have}$   $\langle \text{guarded } ((\nu x) P) \mathbf{by}$  simp
      ultimately have  $\Psi \triangleright \langle \text{Cases } (\text{map } (\lambda(\varphi, P). (\varphi, (\nu x) P)) Cs) \mapsto \alpha \prec$ 
       $(\nu x) P' \mathbf{by}$ (rule Case)
      moreover have  $\langle \Psi, (\nu x) P', (\nu x) P' \rangle \in \text{Rel} \mathbf{by}$ (rule C1)
      ultimately show ?case by blast
    qed
  next
  case(cBrClose  $M$   $xvec$   $N$   $P'$ )

```

```

then show ?case
proof(induct rule: caseCases)
  case(cCase  $\varphi$   $P$ )
  from  $\langle \Psi \triangleright P \mapsto M(\nu*xvec)\langle N \rangle \prec P' \rangle \langle x \# \Psi \rangle \langle x \in \text{supp } M \rangle$ 
  have  $\Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)((\nu*xvec)P')$ 
  by(intro BrClose)
  moreover from  $\langle (\varphi, P) \in \text{set } Cs \rangle$  have  $(\varphi, (\nu x)P) \in \text{set } (\text{map } (\lambda(\varphi, P).$ 
 $(\varphi, (\nu x)P)) Cs)$ 
  by(induct Cs) auto
  moreover from  $\langle \text{guarded } P \rangle$  have guarded(( $\nu x$ ) $P$ ) by simp
  moreover note  $\langle \Psi \vdash \varphi \rangle$ 
  ultimately have  $\Psi \triangleright \text{Cases } \text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs \mapsto \tau \prec$ 
 $(\nu x)((\nu*xvec)P')$ 
  by(intro Case)
  moreover have  $(\Psi, (\nu x)((\nu*xvec)P'), (\nu x)((\nu*xvec)P')) \in \text{Rel}$ 
  by fact
  ultimately show ?case
  by blast
qed
qed
qed
qed

```

lemma *resInputCases*[*consumes 5, case-names cRes*]:

```

fixes  $\Psi$    :: 'b
and  $x$      :: name
and  $P$      :: ('a, 'b, 'c) psi
and  $M$      :: 'a
and  $N$      :: 'a
and  $P'$     :: ('a, 'b, 'c) psi
and  $C$      :: 'd::fs-name

```

```

assumes Trans:  $\Psi \triangleright (\nu x)P \mapsto M(\nu N) \prec P'$ 
and  $x \# \Psi$ 
and  $x \# M$ 
and  $x \# N$ 
and  $x \# P'$ 
and rScope:  $\bigwedge P'. \llbracket \Psi \triangleright P \mapsto M(\nu N) \prec P' \rrbracket \implies \text{Prop } ((\nu x)P')$ 

```

shows *Prop* P'

```

proof –
  note Trans  $\langle x \# \Psi \rangle$ 
  moreover from  $\langle x \# M \rangle \langle x \# N \rangle$  have  $x \# M(\nu N)$  by simp
  moreover note  $\langle x \# P' \rangle$ 
  ultimately show ?thesis using assms
  by(induct rule: resInputCases') simp
qed

```

lemma *resBrInputCases*[*consumes 5, case-names cRes*]:

```

fixes  $\Psi$   :: 'b
and  $x$     :: name
and  $P$     :: ('a, 'b, 'c) psi
and  $M$     :: 'a
and  $N$     :: 'a
and  $P'$    :: ('a, 'b, 'c) psi
and  $C$     :: 'd::fs-name

assumes Trans:  $\Psi \triangleright (\nu x)P \mapsto_{\iota M(N)} \prec P'$ 
and  $x \# \Psi$ 
and  $x \# M$ 
and  $x \# N$ 
and  $x \# P'$ 
and rScope:  $\bigwedge P'. [\Psi \triangleright P \mapsto_{\iota M(N)} \prec P'] \implies Prop ((\nu x)P)$ 

shows Prop P'
proof –
  note Trans  $\langle x \# \Psi \rangle$ 
  moreover from  $\langle x \# M \rangle \langle x \# N \rangle$  have  $x \# \iota M(N)$  by simp
  moreover note  $\langle x \# P' \rangle$ 
  ultimately show ?thesis using assms
    by(induct rule: resBrInputCases') simp
qed

lemma swap-supp:
  fixes  $x$   :: name
  and  $y$     :: name
  and  $N$     :: 'a

assumes  $y \in \text{supp } N$ 

shows  $x \in \text{supp } ([x, y] \cdot N)$ 
  using assms
  by (metis fresh-bij fresh-def swap-simps(2))

lemma swap-supp':
  fixes  $x$   :: name
  and  $y$     :: name
  and  $N$     :: 'a

assumes  $x \in \text{supp } N$ 

shows  $y \in \text{supp } ([x, y] \cdot N)$ 
  using assms
  by (metis fresh-bij fresh-def swap-simps(1))

lemma brOutputFreshSubjectChain:
  fixes  $\Psi$   :: 'b
  and  $Q$     :: ('a, 'b, 'c) psi

```

```

and  $M :: 'a$ 
and  $xvec :: \text{name list}$ 
and  $N :: 'a$ 
and  $Q' :: ('a, 'b, 'c) \text{psi}$ 
and  $yvec :: \text{name list}$ 

assumes  $\Psi \triangleright Q \mapsto \text{iM}(\nu * xvec)\langle N \rangle \prec Q'$ 
and  $xvec \#* M$ 
and  $yvec \#* Q$ 

shows  $yvec \#* M$ 
using  $\text{assms}$ 
proof ( $\text{induct } yvec$ )
  case  $\text{Nil}$ 
  then show  $?case$  by  $\text{simp}$ 
next
  case ( $\text{Cons } a \ yvec$ )
  then have  $yvec \#* M$  by  $\text{simp}$ 
  from  $\langle a \# yvec \rangle \#* Q$  have  $a \# Q$  by  $\text{simp}$ 
  from  $\langle xvec \#* M \rangle \langle a \# Q \rangle \langle \Psi \triangleright Q \mapsto \text{iM}(\nu * xvec)\langle N \rangle \prec Q' \rangle$ 
  have  $a \# M$ 
  by ( $\text{simp add: } \text{brOutputFreshSubject}$ )
  with  $\langle yvec \#* M \rangle$  show  $?case$ 
  by  $\text{simp}$ 
qed

lemma  $\text{scopeExtLeft}$ :
  fixes  $x :: \text{name}$ 
  and  $P :: ('a, 'b, 'c) \text{psi}$ 
  and  $\Psi :: 'b$ 
  and  $Q :: ('a, 'b, 'c) \text{psi}$ 
  and  $\text{Rel} :: ('b \times ('a, 'b, 'c) \text{psi} \times ('a, 'b, 'c) \text{psi}) \text{ set}$ 

assumes  $x \# P$ 
and  $x \# \Psi$ 
and  $\text{eqvt } \text{Rel}$ 
and  $C1: \bigwedge \Psi' R. (\Psi', R, R) \in \text{Rel}$ 
and  $C2: \bigwedge y \Psi' R S \text{zvec}. \llbracket y \# \Psi'; y \# R; \text{zvec} \#* \Psi \rrbracket \implies (\Psi', (\nu y)((\nu * \text{zvec})(R \parallel S)), (\nu * \text{zvec})(R \parallel (\nu y)S)) \in \text{Rel}$ 
and  $C3: \bigwedge \Psi' \text{zvec } R y. \llbracket y \# \Psi'; \text{zvec} \#* \Psi \rrbracket \implies (\Psi', (\nu y)((\nu * \text{zvec})R), (\nu * \text{zvec})(\nu y)R)) \in \text{Rel}$ 
  — Addition for Broadcast
and  $C4: \bigwedge \Psi' R S \text{zvec}. \llbracket \text{zvec} \#* R; \text{zvec} \#* \Psi \rrbracket \implies (\Psi', ((\nu * \text{zvec})(R \parallel S)), (R \parallel ((\nu * \text{zvec})S))) \in \text{Rel}$ 

shows  $\Psi \triangleright (\nu x)(P \parallel Q) \rightsquigarrow_{[\text{Rel}]} P \parallel (\nu x)Q$ 
proof —
  note  $\langle \text{eqvt } \text{Rel} \rangle \langle x \# \Psi \rangle$ 
  moreover have  $x \# (\nu x)(P \parallel Q)$  by ( $\text{simp add: } \text{abs-fresh}$ )

```

moreover from $\langle x \# P \rangle$ **have** $x \# P \parallel (\nu x)Q$ **by** (*simp add: abs-fresh*)
ultimately show *?thesis*
proof (*induct rule: simIFresh[of - - - - x]*)
 case (*cSim* α PQ)
 from $\langle x \# \alpha \rangle \langle bn \alpha \#* (P \parallel (\nu x)Q) \rangle$ **have** $bn \alpha \#* Q$ **by** *simp*
 note $\langle \Psi \triangleright P \parallel (\nu x)Q \mapsto \alpha \prec PQ \rangle \langle bn \alpha \#* \Psi \rangle$
 moreover from $\langle bn \alpha \#* (P \parallel (\nu x)Q) \rangle$ **have** $bn \alpha \#* P$ **and** $bn \alpha \#* (\nu x)Q$
by *simp+*
 ultimately show *?case using* $\langle bn \alpha \#* \text{subject } \alpha \rangle \langle x \# PQ \rangle$
 proof (*induct rule: parCases[where C=x]*)
 case (*cPar1* $P' A_Q \Psi_Q$)
 from $\langle x \# P' \parallel (\nu x)Q \rangle$ **have** $x \# P'$ **by** *simp*
 have $PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'$ **by** *fact*
 from $\langle \text{extractFrame}((\nu x)Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $Fr x Q: (\nu x)(\text{extractFrame } Q)$
 $= \langle A_Q, \Psi_Q \rangle$ **by** *simp*
 then obtain $y A_Q'$ **where** $A: A_Q = y \# A_Q'$ **by** (*cases A_Q*) *auto*
 with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* \alpha \rangle$
 have $A_Q' \#* \Psi$ **and** $A_Q' \#* P$ **and** $A_Q' \#* \alpha$
 and $y \# \Psi$ **and** $y \# P$ **and** $y \# \alpha$
 by *simp+*
 from $PTrans \langle y \# P \rangle \langle y \# \alpha \rangle \langle bn \alpha \#* \text{subject } \alpha \rangle \langle \text{distinct}(bn \alpha) \rangle$ **have** $y \# P'$
 by (*auto intro: freeFreshDerivative*)
 note $PTrans$
 moreover from $A \langle A_Q \#* x \rangle Fr x Q$ **have** $\text{extractFrame}([(y, x)] \cdot Q) = \langle A_Q',$
 $\Psi_Q \rangle$
 by (*simp add: frame.inject alpha' fresh-list-cons eqts*)
 moreover from $\langle bn \alpha \#* Q \rangle$ **have** $([(y, x)] \cdot (bn \alpha)) \#* ([(y, x)] \cdot Q)$
 by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
 with $\langle x \# \alpha \rangle \langle A_Q \#* \alpha \rangle A$ **have** $bn \alpha \#* ([(y, x)] \cdot Q)$ **by** *simp*
 ultimately have $\Psi \triangleright P \parallel ([(y, x)] \cdot Q) \mapsto \alpha \prec (P' \parallel ([(y, x)] \cdot Q))$
 using $\langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle A_Q' \#* \alpha \rangle$
 by (*rule Par1*)
 then have $\Psi \triangleright (\nu y)(P \parallel ([(y, x)] \cdot Q)) \mapsto \alpha \prec (\nu y)(P' \parallel ([(y, x)] \cdot Q))$
 using $\langle y \# \Psi \rangle \langle y \# \alpha \rangle$
 by (*rule Scope*)
 then have $([(y, x)] \cdot \Psi) \triangleright ([(y, x)] \cdot ((\nu y)(P \parallel ([(y, x)] \cdot Q)))) \mapsto ([(y, x)] \cdot$
 $\cdot (\alpha \prec (\nu y)(P' \parallel ([(y, x)] \cdot Q))))$
 by (*rule semantics.eqvt*)
 with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# P \rangle \langle y \# P \rangle \langle x \# \alpha \rangle \langle y \# \alpha \rangle \langle x \# P' \rangle \langle y \# P' \rangle$
 have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \alpha \prec (\nu x)(P' \parallel Q)$
 by (*simp add: eqvts calc-atm*)
 moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle$ **have** $(\Psi, (\nu x)((\nu^* \square)(P' \parallel Q)), (\nu^* \square)(P'$
 $\parallel (\nu x)Q)) \in Rel$
 by (*rule C2*) *auto*
 ultimately show *?case*
 by *force*
next
 case (*cPar2* $xQ' A_P \Psi_P$)
 from $\langle A_P \#* (\nu x)Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* Q$ **by** *simp*

note $\langle \Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto \alpha \prec xQ' \rangle$
moreover have $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by fact**
with $\langle x \# P \rangle \langle A_P \#* x \rangle$ **have** $x \# \Psi_P$ **and** $x \# A_P$
by(force dest: extractFrameFresh)+
with $\langle x \# \Psi \rangle$ **have** $x \# \Psi \otimes \Psi_P$ **by force**
moreover note $\langle x \# \alpha \rangle$
moreover from $\langle x \# P \parallel xQ' \rangle$ **have** $x \# xQ'$ **by simp**
moreover from $FrP \langle bn \alpha \#* P \rangle \langle A_P \#* \alpha \rangle$ **have** $bn \alpha \#* \Psi_P$
by(auto dest: extractFrameFreshChain)
with $\langle bn \alpha \#* \Psi \rangle$ **have** $bn \alpha \#* (\Psi \otimes \Psi_P)$ **by force**
ultimately show ?case **using** $\langle bn \alpha \#* Q \rangle \langle bn \alpha \#* subject \alpha \rangle \langle x \# \alpha \rangle \langle bn \alpha \#* P \rangle \langle A_P \#* \alpha \rangle \langle bn \alpha \#* \Psi \rangle \langle x \# P \rangle$
proof(induct rule: resCases'[where $C=(P, A_P, \Psi)$])
case(cOpen $M \ xvec1 \ xvec2 \ y \ N \ Q'$)
from $\langle x \# M(\nu*(xvec1@y\#xvec2))\langle N \rangle \rangle$ **have** $x \# xvec1$ **and** $x \neq y$ **and** $x \# xvec2$ **by simp+**
from $\langle bn(M(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* \Psi \rangle$ **have** $y \# \Psi$ **by simp**
note $\langle \Psi \otimes \Psi_P \triangleright ([x, y] \cdot Q) \mapsto M(\nu*(xvec1@xvec2))\langle N \rangle \prec Q' \rangle$ FrP
moreover from $\langle bn(M(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* P \rangle$ **have** $(xvec1@xvec2) \#* P$ **and** $y \# P$ **by simp+**
moreover from $\langle A_P \#* (M(\nu*(xvec1@y\#xvec2))\langle N \rangle) \rangle$ **have** $A_P \#* (M(\nu*(xvec1@xvec2))\langle N \rangle)$ **and** $y \# A_P$ **by simp+**
moreover from $\langle A_P \#* Q \rangle \langle x \# A_P \rangle \langle y \# A_P \rangle$ **have** $A_P \#* ([x, y] \cdot Q)$
by simp
ultimately have $\Psi \triangleright P \parallel ([x, y] \cdot Q) \mapsto M(\nu*(xvec1@xvec2))\langle N \rangle \prec (P \parallel Q')$
using $\langle A_P \#* \Psi \rangle$
by(intro Par2) (assumption | simp)+
then have $\Psi \triangleright (\nu y)(P \parallel ([x, y] \cdot Q)) \mapsto M(\nu*(xvec1@y\#xvec2))\langle N \rangle$
 $\prec (P \parallel Q')$
using $\langle y \in supp N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$
by(rule Open)
with $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$ **have** $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto M(\nu*(xvec1@y\#xvec2))\langle N \rangle$
 $\prec (P \parallel Q')$
by(subst alphaRes[where $y=y$]) (simp add: fresh-left calc-atm eqvts)+
moreover have $(\Psi, P \parallel Q', P \parallel Q') \in Rel$ **by**(rule C1)
ultimately show ?case **by blast**
next
case(cBrOpen $M \ xvec1 \ xvec2 \ y \ N \ Q'$)
from $\langle x \# iM(\nu*(xvec1@y\#xvec2))\langle N \rangle \rangle$ **have** $x \# xvec1$ **and** $x \neq y$ **and** $x \# xvec2$ **by simp+**
from $\langle bn(iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* \Psi \rangle$ **have** $y \# \Psi$ **by simp**
note $\langle \Psi \otimes \Psi_P \triangleright ([x, y] \cdot Q) \mapsto iM(\nu*(xvec1@xvec2))\langle N \rangle \prec Q' \rangle$ FrP
moreover from $\langle bn(iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \#* P \rangle$ **have** $(xvec1@xvec2) \#* P$ **and** $y \# P$ **by simp+**
moreover from $\langle A_P \#* (iM(\nu*(xvec1@y\#xvec2))\langle N \rangle) \rangle$ **have** $A_P \#* (iM(\nu*(xvec1@xvec2))\langle N \rangle)$ **and** $y \# A_P$ **by simp+**
moreover from $\langle A_P \#* Q \rangle \langle x \# A_P \rangle \langle y \# A_P \rangle$ **have** $A_P \#* ([x, y] \cdot Q)$
by simp

```

    ultimately have  $\Psi \triangleright P \parallel ((x, y) \cdot Q) \mapsto_{iM} (\nu^*(xvec1 @ xvec2)) \langle N \rangle \prec$ 
  (P || Q')
    using  $\langle A_P \# \Psi \rangle$ 
    by(intro Par2) (assumption | simp)+
    then have  $\Psi \triangleright (\nu y)(P \parallel ((x, y) \cdot Q)) \mapsto_{iM} (\nu^*(xvec1 @ y \# xvec2)) \langle N \rangle$ 
  < (P || Q')
    using  $\langle y \in \text{supp } N \rangle \langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ 
    by(rule BrOpen)
  with  $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$  have  $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_{iM} (\nu^*(xvec1 @ y \# xvec2)) \langle N \rangle$ 
  < (P || Q')
    by(subst alphaRes[where y=y]) (simp add: fresh-left calc-atm eqvts)+
    moreover have  $(\Psi, P \parallel Q', P \parallel Q') \in \text{Rel}$  by(rule C1)
    ultimately show ?case by blast
  next
  case(cRes Q')
  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle$  FrP  $\langle \text{bn } \alpha \# P \rangle$ 
  have  $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$  using  $\langle A_P \# \Psi \rangle \langle A_P \# Q \rangle \langle A_P \# \alpha \rangle$ 
  by(rule Par2)
  then have  $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \alpha \prec (\nu x)(P \parallel Q')$  using  $\langle x \# \Psi \rangle \langle x \# \alpha \rangle$ 
  by(rule Scope)
  moreover from  $\langle x \# \Psi \rangle \langle x \# P \rangle$  have  $(\Psi, (\nu x)((\nu^*[]) (P \parallel Q')), (\nu^*[]) (P$ 
  ||  $(\nu x) Q')) \in \text{Rel}$ 
  by(rule C2) auto
  ultimately show ?case
  by force
  next
  case(cBrClose M xvec N Q')
  from  $\langle xvec \# (P, A_P, \Psi) \rangle$ 
  have  $xvec \# P$  and  $A_P \# xvec$  and  $xvec \# \Psi$ 
  by simp+
  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec Q' \rangle \langle A_P \# Q \rangle$ 
   $\langle xvec \# M \rangle$ 
  have  $A_P \# M$ 
  by(simp add: brOutputFreshSubjectChain)

  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec Q' \rangle \langle xvec \# M \rangle \langle \text{distinct } xvec \rangle$ 
   $\langle A_P \# Q \rangle \langle A_P \# xvec \rangle$ 
  have  $A_P \# N$  and  $A_P \# Q'$  by(simp add: brOutputFreshChainDerivative)+
  from  $\langle A_P \# M \rangle \langle A_P \# xvec \rangle \langle A_P \# N \rangle$  have  $A_P \# (iM(\nu^* xvec) \langle N \rangle)$ 
  by simp

  from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec Q' \rangle \langle \text{extractFrame } P = \langle A_P,$ 
   $\Psi_P \rangle \rangle \langle xvec \# P \rangle \langle A_P \# \Psi \rangle \langle A_P \# Q \rangle \langle A_P \# (iM(\nu^* xvec) \langle N \rangle) \rangle$ 
  have  $\Psi \triangleright P \parallel Q \mapsto_{iM} (\nu^* xvec) \langle N \rangle \prec P \parallel Q'$ 
  by(simp add: Par2)

  from  $\langle x \# \Psi \rangle \langle x \# P \rangle \langle xvec \# P \rangle \langle xvec \# \Psi \rangle$ 
  have  $(x \# xvec) \# P$  and  $(x \# xvec) \# \Psi$  by simp+
  then have  $(\Psi, ((\nu^*(x \# xvec))(P \parallel Q')), P \parallel ((\nu^*(x \# xvec)) Q')) \in \text{Rel}$ 

```


by(rule C_4)
then have $(\Psi, (\nu x)((\nu * xvec)(P \parallel Q')), P \parallel (\nu x)((\nu * xvec)Q') \in Rel$
by simp

moreover from $\langle \Psi \triangleright P \parallel Q \mapsto \downarrow M(\nu * xvec)\langle N \rangle \prec P \parallel Q' \rangle \langle x \in supp M \rangle$
 $\langle x \# \Psi \rangle$
have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \tau \prec (\nu x)((\nu * xvec)(P \parallel Q'))$
by(rule $BrClose$)

ultimately show $?case$
by force

qed
next

case($cComm1 \Psi_Q M N P' A_P \Psi_P K xvec xQ' A_Q$)
have $QTrans: \Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto K(\nu * xvec)\langle N \rangle \prec xQ'$ **and** $FrQ: extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle$ **by** $fact+$
have $PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto M\langle N \rangle \prec P'$ **and** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** $fact+$
have $x \# (\nu x)Q$ **by**($simp$ $add: abs-fresh$)
with $QTrans$ **have** $x \# N$ **and** $x \# xQ'$ **using** $\langle xvec \#* x \rangle \langle xvec \#* K \rangle \langle distinct xvec \rangle$
by($force$ $intro: outputFreshDerivative$) $+$
from $PTrans \langle x \# P \rangle \langle x \# N \rangle$ **have** $x \# P'$ **by**(rule $inputFreshDerivative$)
from $\langle x \# (\nu x)Q \rangle FrQ \langle A_Q \#* x \rangle$ **have** $x \# \Psi_Q$
by($drule-tac extractFrameFresh$) $auto$
from $\langle x \# P \rangle FrP \langle A_P \#* x \rangle$ **have** $x \# \Psi_P$
by($drule-tac extractFrameFresh$) $auto$
from $\langle A_P \#* (\nu x)Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* Q$ **by** $simp$
from $\langle A_Q \#* (\nu x)Q \rangle \langle A_Q \#* x \rangle$ **have** $A_Q \#* Q$ **by** $simp$
from $PTrans FrP \langle distinct A_P \rangle \langle x \# P \rangle \langle A_Q \#* P \rangle \langle xvec \#* P \rangle \langle A_P \#* \Psi \rangle$
 $\langle A_P \#* \Psi_Q \rangle \langle A_P \#* x \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_P \#* xvec \rangle$
obtain M' **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \# M'$ **and** $A_Q \#* M'$
and $xvec \#* M'$
by($elim$ $inputObtainPrefix$ [**where** $B=x\#xvec@A_Q$]) ($assumption$ | $force$ $simp$ $add: fresh-star-list-cons$) $+$
then have $MeqM': \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$ **by**($metis$ $statEqEnt$ $Associativity$ $Commutativity$ $Composition$)
with $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by($blast$ $intro: chanEqTrans$ $chanEqSym$)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$ **by**($metis$ $statEqEnt$ $Associativity$ $Commutativity$ $Composition$)
with $QTrans FrQ \langle distinct A_Q \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* ((\nu x)Q) \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$
have $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto M'(\nu * xvec)\langle N \rangle \prec xQ'$
by($force$ $intro: outputRenameSubject$)
moreover from $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$ **have** $x \# \Psi \otimes \Psi_P$ **by** $force$
moreover from $\langle xvec \#* x \rangle$ **have** $x \# xvec$ **by** $simp$
with $\langle x \# M' \rangle \langle x \# N \rangle$ **have** $x \# M'(\nu * xvec)\langle N \rangle$ **by** $simp$
moreover note $\langle x \# xQ' \rangle$

moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ **have** $xvec \#* (\Psi \otimes \Psi_P)$ **by force**
moreover from $\langle xvec \#* (\nu x) Q \rangle \langle x \# xvec \rangle$ **have** $xvec \#* Q$ **by simp**
moreover note $\langle xvec \#* M' \rangle$

moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ **have** $bn(M'(\nu * xvec)\langle N \rangle) \#* (\Psi \otimes \Psi_P)$ **by force**
moreover from $\langle xvec \#* (\nu x) Q \rangle \langle x \# xvec \rangle$ **have** $bn(M'(\nu * xvec)\langle N \rangle) \#* Q$
by simp
moreover from $\langle xvec \#* P \rangle$ **have** $bn(M'(\nu * xvec)\langle N \rangle) \#* P$ **by simp**
from $\langle xvec \#* \Psi \rangle$ **have** $bn(M'(\nu * xvec)\langle N \rangle) \#* \Psi$ **by simp**
from $\langle A_Q \#* xvec \rangle \langle A_Q \#* M' \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* (M'(\nu * xvec)\langle N \rangle)$ **by simp**
have $object(M'(\nu * xvec)\langle N \rangle) = Some\ N$ **by simp**
have $bn(M'(\nu * xvec)\langle N \rangle) = xvec$ **by simp**
have $subject(M'(\nu * xvec)\langle N \rangle) = Some\ M'$ **by simp**
from $\langle xvec \#* M' \rangle$ **have** $bn(M'(\nu * xvec)\langle N \rangle) \#* subject(M'(\nu * xvec)\langle N \rangle)$ **by simp**

ultimately show ?case
using $\langle x \# M'(\nu * xvec)\langle N \rangle \rangle \langle bn(M'(\nu * xvec)\langle N \rangle) \#* P \rangle \langle bn(M'(\nu * xvec)\langle N \rangle) \#* \Psi \rangle \langle object(M'(\nu * xvec)\langle N \rangle) = Some\ N \rangle$
 $\langle bn(M'(\nu * xvec)\langle N \rangle) = xvec \rangle \langle subject(M'(\nu * xvec)\langle N \rangle) = Some\ M' \rangle \langle A_Q \#* (M'(\nu * xvec)\langle N \rangle) \rangle$

proof(induct rule: resOutputCases''''')
case(cOpen M'' xvec1 xvec2 y N' Q')
then have $Eq: M'(\nu * xvec)\langle N \rangle = M''(\nu * (xvec1 @ y \# xvec2))\langle N \rangle$ **by simp**
from $\langle x \# M'(\nu * xvec)\langle N \rangle \rangle Eq$ **have** $x \# xvec1$ **and** $x \neq y$ **and** $x \# xvec2$ **and** $x \# M''$
by simp+
from $\langle bn(M'(\nu * xvec)\langle N \rangle) \#* P \rangle Eq$ **have** $(xvec1 @ xvec2) \#* P$ **and** $y \# P$ **by simp+**
from $\langle A_Q \#* (M'(\nu * xvec)\langle N \rangle) \rangle Eq$ **have** $(xvec1 @ xvec2) \#* A_Q$ **and** $y \# A_Q$ **and** $A_Q \#* M''$ **by simp+**
from $\langle bn(M'(\nu * xvec)\langle N \rangle) \#* \Psi \rangle Eq$ **have** $(xvec1 @ xvec2) \#* \Psi$ **and** $y \# \Psi$ **by simp+**
from Eq **have** $N = N'$ **and** $xvec = xvec1 @ y \# xvec2$ **and** $M' = M''$ **by(simp add: action.inject)+**
from $\langle x \# P \rangle \langle y \# P \rangle \langle x \neq y \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle N \rangle) \prec P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle [(y, x)] \cdot N \rangle) \prec \langle [(y, x)] \cdot P' \rangle$
by(elim inputAlpha[where xvec=[y]]) (auto simp add: calc-atm)
then have $PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle [(x, y)] \cdot N \rangle) \prec \langle [(x, y)] \cdot P' \rangle$
by(simp add: name-swap)

from $\langle \Psi \otimes \Psi_P \triangleright [(x, y)] \cdot Q \mapsto M''(\nu * (xvec1 @ xvec2))\langle N \rangle \prec Q' \rangle$
have $\langle [(x, y)] \cdot (\Psi \otimes \Psi_P \triangleright [(x, y)] \cdot Q) \mapsto M''(\nu * (xvec1 @ xvec2))\langle N \rangle \prec Q' \rangle$
by simp
then have $\langle [(x, y)] \cdot (\Psi \otimes \Psi_P) \triangleright \langle [(x, y)] \cdot \langle [(x, y)] \cdot Q \rangle \rangle \mapsto \langle [(x, y)] \cdot Q' \rangle$

$M''(\nu^*([(x, y)] \cdot (xvec1 @ xvec2))) \langle [(x, y)] \cdot N' \rangle \prec [(x, y)] \cdot Q'$
by (*simp add: eqvts*)
with $\langle x \# (\Psi \otimes \Psi_P) \rangle \langle y \# (\Psi \otimes \Psi_P) \rangle \langle x \# xvec1 \rangle \langle y \# xvec1 \rangle \langle x \# xvec2 \rangle$
 $\langle y \# xvec2 \rangle \langle x \# M'' \rangle \langle y \# M'' \rangle$
have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto M''(\nu^*(xvec1 @ xvec2)) \langle [(x, y)] \cdot N' \rangle \prec$
 $[(x, y)] \cdot Q'$
by *simp*
with $\langle A_Q \# x \rangle \langle y \# A_Q \rangle \langle distinct\ xvec1 \rangle \langle distinct\ xvec2 \rangle \langle xvec1 \# xvec2 \rangle$
 $\langle xvec1 \# M'' \rangle \langle xvec2 \# M'' \rangle$
 $\langle (xvec1 @ xvec2) \# A_Q \rangle$
have $A_Q \# [(x, y)] \cdot Q'$ **using** $\langle A_Q \# Q \rangle$
by (*elim outputFreshChainDerivative(2)*) (*assumption | simp*)

from $\langle extractFrame(\nu x) Q \rangle = \langle A_Q, \Psi_Q \rangle$ **have** $FrQ: (\nu x)(extractFrame$
 $Q) = \langle A_Q, \Psi_Q \rangle$ **by** *simp*
then obtain $z A_Q'$ **where** $A: A_Q = z \# A_Q'$ **by** (*cases A_Q*) *auto*
with $\langle A_Q \# \Psi \rangle \langle A_Q \# P \rangle \langle A_Q \# P' \rangle \langle A_Q \# \Psi_P \rangle \langle A_Q \# Q \rangle \langle (xvec1 @ xvec2)$
 $\# A_Q \rangle \langle A_Q \# M'' \rangle \langle A_Q \# [(x, y)] \cdot Q' \rangle \langle y \# A_Q \rangle \langle A_Q \# N \rangle$
have $A_Q' \# \Psi$ **and** $A_Q' \# P$ **and** $A_Q' \# \Psi_P$ **and** $A_Q' \# Q$
and $z \# \Psi$ **and** $z \# P$ **and** $z \# P'$ **and** $z \# \Psi_P$ **and** $z \# Q$ **and** $z \# xvec1$
and $z \# xvec2$
and $z \# M''$ **and** $z \# [(x, y)] \cdot Q'$ **and** $A_Q' \# M''$ **and** $z \neq y$ **and** $z \#$
 $(xvec1 @ xvec2)$
by *auto*
from $A \langle A_P \# A_Q \rangle$ **have** $A_P \# A_Q'$ **and** $z \# A_P$ **by** *simp+*
from $A \langle A_Q \# x \rangle$ **have** $x \neq z$ **and** $x \# A_Q'$ **by** *simp+*

from $\langle distinct\ A_Q \rangle A$ **have** $z \# A_Q'$
by (*induct A_Q'*) (*auto simp add: fresh-list-nil fresh-list-cons*)
from $PTrans \langle x \# P \rangle \langle z \# P \rangle \langle x \neq z \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle [(x, z)] \cdot$
 $[(x, y)] \cdot N \rangle) \prec (\langle [(x, z)] \cdot [(x, y)] \cdot P' \rangle)$
by (*elim inputAlpha[where xvec=[x]]*) (*auto simp add: calc-atm*)
moreover note FrP
moreover from $QTrans$ **have** $(\langle [(x, z)] \cdot (\Psi \otimes \Psi_P) \rangle \triangleright \langle [(x, z)] \cdot Q \rangle \mapsto \langle [(x,$
 $z) \rangle \cdot (M''(\nu^*(xvec1 @ xvec2)) \langle [(x, y)] \cdot N' \rangle) \prec \langle [(x, y)] \cdot Q' \rangle)$
by (*rule semantics.eqt*)
with $\langle x \# \Psi \rangle \langle z \# \Psi \rangle \langle x \# \Psi_P \rangle \langle z \# \Psi_P \rangle \langle x \# M'' \rangle \langle z \# M'' \rangle \langle x \# xvec1 \rangle \langle x \#$
 $xvec2 \rangle \langle z \# xvec1 \rangle \langle z \# xvec2 \rangle$
have $\Psi \otimes \Psi_P \triangleright (\langle [(x, z)] \cdot Q \rangle \mapsto M''(\nu^*(xvec1 @ xvec2)) \langle [(x, z)] \cdot [(x, y)]$
 $\cdot N' \rangle) \prec (\langle [(x, z)] \cdot [(x, y)] \cdot Q' \rangle)$
by (*simp add: eqvts*)
moreover from $A \langle A_Q \# x \rangle FrQ$ **have** $extractFrame(\langle [(x, z)] \cdot Q \rangle) = \langle A_Q',$
 $\Psi_Q \rangle$
by (*clarsimp simp add: alpha' eqvts frame.inject fresh-list-cons name-swap*)
moreover from $\langle A_P \# Q \rangle$ **have** $(\langle [(x, z)] \cdot A_P \rangle \# \langle [(x, z)] \cdot Q \rangle)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \# x \rangle \langle z \# A_P \rangle$ **have** $A_P \# (\langle [(x, z)] \cdot Q \rangle)$ **by** *simp*
moreover from $\langle A_Q' \# Q \rangle$ **have** $(\langle [(x, z)] \cdot A_Q' \rangle \# \langle [(x, z)] \cdot Q \rangle)$
by (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle x \# A_Q' \rangle \langle z \# A_Q' \rangle$ **have** $A_Q' \#* ((x, z) \cdot Q)$ **by** *simp*
ultimately have $\Psi \triangleright (P \parallel ((x, z) \cdot Q)) \mapsto \tau \prec (\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q'])))$
using $MeqM' \langle M'=M'' \rangle \langle N=N' \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* A_Q' \rangle \langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle (xvec1@xvec2) \#* P \rangle \langle A_P \#* M \rangle \langle A_Q' \#* M'' \rangle$
by (*intro Comm1*) (*assumption* | *simp*) +
with $\langle z \# \Psi \rangle$ **have** $\Psi \triangleright (\nu z)(P \parallel ((x, z) \cdot Q)) \mapsto \tau \prec (\nu z)((\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q'])))$
by (*intro Scope*) *auto*
moreover from $\langle x \# P \rangle \langle z \# P \rangle \langle z \# Q \rangle$ **have** $(\nu z)(P \parallel ((x, z) \cdot Q)) = (\nu x)((x, z) \cdot (P \parallel ((x, z) \cdot Q)))$
by (*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)
with $\langle x \# P \rangle \langle z \# P \rangle$ **have** $(\nu z)(P \parallel ((x, z) \cdot Q)) = (\nu x)(P \parallel Q)$
by (*simp add: eqvts*)
moreover from $\langle z \neq y \rangle \langle x \neq z \rangle \langle z \# P' \rangle \langle z \# [(x, y) \cdot Q'] \rangle$ **have** $(\nu z)((\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q']))) = (\nu x)((x, z) \cdot ((\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q']))))$
by (*subst alphaRes[of x]*) (*auto simp add: resChainFresh fresh-left calc-atm name-swap*)
with $\langle x \# xvec1 \rangle \langle x \# xvec2 \rangle \langle z \# xvec1 \rangle \langle z \# xvec2 \rangle$ **have** $(\nu z)((\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q']))) = (\nu x)((\nu*(xvec1@xvec2))(((x, z) \cdot [(x, y) \cdot P'] \parallel ((x, y) \cdot Q')))$
by (*simp add: eqvts*)
moreover from $\langle x \# P' \rangle \langle x \# Q' \rangle \langle x \# xvec1 \rangle \langle x \# xvec2 \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$
have $(\nu x)((\nu*(xvec1@xvec2))(((x, y) \cdot P') \parallel ((x, y) \cdot Q'))) = (\nu y)((\nu*(xvec1@xvec2))(P' \parallel Q'))$
by (*subst alphaRes[of y]*) (*auto simp add: resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \tau \prec (\nu y)((\nu*(xvec1@xvec2))(P' \parallel Q'))$
by *simp*
moreover from $\langle y \# \Psi \rangle \langle (xvec1@xvec2) \#* \Psi \rangle \langle xvec=xvec1@y\#xvec2 \rangle$
have $(\Psi, (\nu y)((\nu*(xvec1@xvec2))(P' \parallel Q')), (\nu*xvec)(P' \parallel Q')) \in Rel$
by (*force intro: C3 simp add: resChainAppend*)
ultimately show *?case by blast*
next
case (*cRes Q'*)
have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto M'(\nu*xvec)\langle N \rangle \prec Q'$ **by** *fact*
with $\langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle \langle xvec \#* M' \rangle \langle distinct\ xvec \rangle$ **have** $A_Q \#* Q'$
by (*force dest: outputFreshChainDerivative*)

with $\langle extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $FrxQ: (\nu x)(extractFrame Q) = \langle A_Q, \Psi_Q \rangle$ **by** *simp*
then obtain $y A_Q'$ **where** $A: A_Q = y\#A_Q'$ **by** (*cases A_Q*) *auto*
with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle \langle A_Q \#* M' \rangle \langle A_Q \#* Q' \rangle$
have $A_Q' \#* \Psi$ **and** $A_Q' \#* P$ **and** $A_Q' \#* \Psi_P$ **and** $A_Q' \#* Q$ **and** $A_Q \#*$

$xvec$ and $A_Q \#* Q'$
 and $y \# \Psi$ and $y \# P$ and $y \# P'$ and $y \# \Psi_P$ and $y \# Q$ and $y \# xvec$
 and $y \# M'$ and $y \# Q'$
 and $A_{Q'} \#* M'$
 by(*simp*) +
 from $A \langle A_P \#* A_Q \rangle$ have $A_P \#* A_{Q'}$ and $y \# A_P$ by(*simp add: fresh-star-list-cons*) +
 from $A \langle A_Q \#* x \rangle$ have $x \neq y$ and $x \# A_{Q'}$ by(*simp add: fresh-list-cons*) +

 with $A \langle \text{distinct } A_Q \rangle$ have $y \# A_{Q'}$
 by(*induct A_Q'*) (*auto simp add: fresh-list-nil fresh-list-cons*)

 from $\langle x \# P \rangle \langle y \# P \rangle \langle x \neq y \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P' \rangle$
 have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle [(y, x)] \cdot N \rangle) \prec [(y, x)] \cdot P'$
 by(*intro inputAlpha[where xvec=[y]]*) (*auto simp add: calc-atm*)
 then have $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\langle [(x, y)] \cdot N \rangle) \prec [(x, y)] \cdot P'$
 by(*simp add: name-swap*)
 moreover note *FrP*
 moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M'(\nu * xvec) \langle N \rangle \prec Q' \rangle$ have $\langle [(x, y)] \cdot (\Psi \otimes \Psi_P) \triangleright \langle [(x, y)] \cdot Q \rangle \mapsto \langle [(x, y)] \cdot (M'(\nu * xvec) \langle N \rangle \prec Q') \rangle$
 by(*rule semantics.eqvt*)
 with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# \Psi_P \rangle \langle y \# \Psi_P \rangle \langle x \# M' \rangle \langle y \# M' \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle$
 have $\Psi \otimes \Psi_P \triangleright \langle [(x, y)] \cdot Q \rangle \mapsto M'(\nu * xvec) \langle \langle [(x, y)] \cdot N \rangle \rangle \prec \langle [(x, y)] \cdot Q' \rangle$
 by(*simp add: eqvts*)
 moreover from $A \langle A_Q \#* x \rangle$ *FrQ* have *FrQ*: *extractFrame*($\langle [(x, y)] \cdot Q \rangle$)
 = $\langle A_{Q'}, \Psi_Q \rangle$
 by(*clarsimp simp add: alpha' eqvts frame.inject fresh-list-cons name-swap*)
 moreover from $\langle A_P \#* Q \rangle$ have $\langle [(x, y)] \cdot A_P \rangle \#* \langle [(x, y)] \cdot Q \rangle$ by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
 with $\langle A_P \#* x \rangle \langle y \# A_P \rangle$ have $A_P \#* \langle [(x, y)] \cdot Q \rangle$ by *simp*
 moreover from $\langle A_{Q'} \#* Q \rangle$ have $\langle [(x, y)] \cdot A_{Q'} \rangle \#* \langle [(x, y)] \cdot Q \rangle$ by(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
 with $\langle x \# A_{Q'} \rangle \langle y \# A_{Q'} \rangle$ have $A_{Q'} \#* \langle [(x, y)] \cdot Q \rangle$ by *simp*
 ultimately have $\Psi \triangleright (P \parallel \langle [(x, y)] \cdot Q \rangle) \mapsto \tau \prec (\nu * xvec) \langle \langle [(x, y)] \cdot P' \rangle \parallel \langle [(x, y)] \cdot Q' \rangle \rangle$
 using *MeqM'* $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* A_{Q'} \rangle \langle A_{Q'} \#* \Psi \rangle \langle A_{Q'} \#* P \rangle$
 $\langle xvec \#* P \rangle \langle A_P \#* M \rangle \langle A_{Q'} \#* M' \rangle$
 by(*intro Comm1*) (*assumption | simp*) +
 with $\langle y \# \Psi \rangle$ have $\Psi \triangleright (\nu y)(P \parallel \langle [(x, y)] \cdot Q \rangle) \mapsto \tau \prec (\nu y)((\nu * xvec) \langle \langle [(x, y)] \cdot P' \rangle \parallel \langle [(x, y)] \cdot Q' \rangle \rangle)$
 by(*intro Scope*) *auto*
 moreover from $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$ have $(\nu y)(P \parallel \langle [(x, y)] \cdot Q \rangle) = (\nu x)(\langle [(x, y)] \cdot (P \parallel \langle [(x, y)] \cdot Q \rangle) \rangle)$
 by(*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)
 with $\langle x \# P \rangle \langle y \# P \rangle$ have $(\nu y)(P \parallel \langle [(x, y)] \cdot Q \rangle) = (\nu x)(P \parallel Q)$
 by(*simp add: eqvts*)
 moreover from $\langle y \# P' \rangle \langle y \# Q' \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle$ have $(\nu y)((\nu * xvec) \langle \langle [(x, y)] \cdot P' \rangle \parallel \langle [(x, y)] \cdot Q' \rangle \rangle) = (\nu x)((\nu * xvec) \langle P' \parallel Q' \rangle)$

by(subst alphaRes[*of y*]) (*auto simp add: resChainFresh calc-atm eqts fresh-left name-swap*)
ultimately have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \tau \prec (\nu x)((\nu * xvec)(P' \parallel Q'))$
by simp
moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle \langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu x)((\nu * xvec)(P' \parallel Q'))), (\nu * xvec)(P' \parallel (\nu x)Q') \in Rel$
by(rule C2)
ultimately show ?*case* **by blast**
qed
next
case(cComm2 $\Psi_Q M xvec N P' A_P \Psi_P K xQ' A_Q$)
have *QTrans*: $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto K(N) \prec xQ'$ **and** *FrQ*: *extractFrame* $((\nu x)Q) = \langle A_Q, \Psi_Q \rangle$ **by fact+**
have *PTrans*: $\Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P'$ **and** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **by fact+**
from *PTrans* $\langle x \# P \rangle$ **have** $x \# N$ **and** $x \# P'$ **using** $\langle xvec \#* x \rangle \langle xvec \#* M \rangle$ *distinct xvec*
by(*force intro: outputFreshDerivative*)+
have $x \# (\nu x)Q$ **by**(*simp add: abs-fresh*)
with *FrQ* $\langle A_Q \#* x \rangle$ **have** $x \# \Psi_Q$
by(*drule-tac extractFrameFresh*) *auto*
from $\langle x \# P \rangle$ *FrP* $\langle A_P \#* x \rangle$ **have** $x \# \Psi_P$
by(*drule-tac extractFrameFresh*) *auto*
from $\langle A_P \#* (\nu x)Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* Q$ **by simp**
from $\langle A_Q \#* (\nu x)Q \rangle \langle A_Q \#* x \rangle$ **have** $A_Q \#* Q$ **by simp**
from $\langle xvec \#* x \rangle \langle xvec \#* (\nu x)Q \rangle$ **have** $xvec \#* Q$ **by simp**

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P' \rangle$ **have** *PResTrans*: $\Psi \otimes \Psi_Q \triangleright P \mapsto ROut M ((\nu * xvec)N \prec' P')$
by(*simp add: residualInject*)

from *PResTrans* *FrP* *distinct A_P* $\langle x \# P \rangle \langle A_Q \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle$
 $\langle A_P \#* x \rangle \langle A_P \#* A_Q \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle xvec \#* M \rangle$ *distinct xvec*
obtain M' **where** $(\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \# M'$ **and** $A_Q \#* M'$
by(*elim outputObtainPrefix[where B=x#A_Q]*) (*assumption | force simp add: fresh-star-list-cons*)+
then have *MeqM'*: $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro: chanEqTrans chanEqSym*)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with *QTrans* *FrQ* *distinct A_Q* $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* ((\nu x)Q) \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$
have $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto M'(\nu x) \prec xQ'$ **by**(*force intro: inputRenameSubject*)

moreover from $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$ **have** $x \# \Psi \otimes \Psi_P$ **by force**
moreover note $\langle x \# M' \rangle \langle x \# N \rangle$
moreover from *QTrans* $\langle x \# N \rangle$ **have** $x \# xQ'$ **by**(*force dest: inputFreshDeriva-*

tive simp add: abs-fresh
ultimately show *?case*
proof(*induct rule: resInputCases*)
case(*cRes Q'*)
have *QTrans*: $\Psi \otimes \Psi_P \triangleright Q \mapsto M'(|N|) \prec Q'$ **by fact**
with $\langle A_Q \#* Q \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* Q'$
by(*elim inputFreshChainDerivative*)

with $\langle \text{extractFrame}(|\nu x|Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** *FrqQ*: $(|\nu x|)(\text{extractFrame } Q) = \langle A_Q, \Psi_Q \rangle$ **by simp**
then obtain $y A_Q'$ **where** $A: A_Q = y\#A_Q'$ **by**(*cases A_Q*) *auto*
with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* \text{vec} \rangle$
 $\langle A_Q \#* M' \rangle \langle A_Q \#* Q' \rangle \langle A_Q \#* N \rangle$
have $A_Q' \#* \Psi$ **and** $A_Q' \#* P$ **and** $A_Q' \#* \Psi_P$ **and** $A_Q' \#* Q$ **and** $A_Q \#*$
 vec **and** $A_Q \#* Q'$
and $y \# \Psi$ **and** $y \# P$ **and** $y \# P'$ **and** $y \# \Psi_P$ **and** $y \# Q$ **and** $y \# \text{vec}$
and $y \# M'$ **and** $y \# Q'$ **and** $y \# N$
and $A_Q' \#* M'$
by(*simp*)
from $A \langle A_P \#* A_Q \rangle$ **have** $A_P \#* A_Q'$ **and** $y \# A_P$ **by**(*simp add: fresh-star-list-cons*)
from $A \langle A_Q \#* x \rangle$ **have** $x \neq y$ **and** $x \# A_Q'$ **by**(*simp add: fresh-list-cons*)

with $A \langle \text{distinct } A_Q \rangle$ **have** $y \# A_Q'$
by(*induct A_Q'*) (*auto simp add: fresh-list-nil fresh-list-cons*)

note *PTrans FrP*
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M'(|N|) \prec Q' \rangle$ **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \triangleright ([(x, y)] \cdot Q) \mapsto ([(x, y)] \cdot (M'(|N|) \prec Q'))$
by(*rule semantics.eqt*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# \Psi_P \rangle \langle y \# \Psi_P \rangle \langle x \# M' \rangle \langle y \# M' \rangle \langle x \# N \rangle \langle y \# N \rangle$
have $\Psi \otimes \Psi_P \triangleright ([(x, y)] \cdot Q) \mapsto M'(|N|) \prec ([(x, y)] \cdot Q')$
by(*simp add: eqts*)
moreover from $A \langle A_Q \#* x \rangle$ *FrqQ* **have** *FrQ*: $\text{extractFrame}([(x, y)] \cdot Q) = \langle A_Q', \Psi_Q \rangle$ **and** $y \# \text{extractFrame } Q$
by(*clarsimp simp add: alpha' eqts frame.inject fresh-list-cons name-swap*)
moreover from $\langle A_P \#* Q \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle y \# A_P \rangle$ **have** $A_P \#* ([(x, y)] \cdot Q)$ **by simp**
moreover from $\langle A_Q' \#* Q \rangle$ **have** $([(x, y)] \cdot A_Q') \#* ([(x, y)] \cdot Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_Q' \rangle \langle y \# A_Q' \rangle$ **have** $A_Q' \#* ([(x, y)] \cdot Q)$ **by simp**
moreover from $\langle \text{vec} \#* Q \rangle$ **have** $([(x, y)] \cdot \text{vec}) \#* ([(x, y)] \cdot Q)$ **by**(*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle \text{vec} \#* x \rangle \langle y \# \text{vec} \rangle$ **have** $\text{vec} \#* ([(x, y)] \cdot Q)$ **by simp**
ultimately have $\Psi \triangleright (P \parallel ([(x, y)] \cdot Q)) \mapsto \tau \prec (|\nu * \text{vec}|)(P' \parallel ([(x, y)] \cdot Q'))$
using *MeqM'* $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* A_Q' \rangle \langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle$
 $\langle A_P \#* M \rangle \langle A_Q' \#* M' \rangle$
by(*intro Comm2*) (*assumption | simp*)

with $\langle y \# \Psi \rangle$ **have** $\Psi \triangleright (\nu y)(P \parallel ((x, y) \cdot Q)) \mapsto \tau \prec (\nu y)((\nu * xvec)(P' \parallel ((x, y) \cdot Q')))$
by (*intro Scope*) *auto*
moreover from $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$ **have** $(\nu y)(P \parallel ((x, y) \cdot Q)) = (\nu x)((x, y) \cdot (P \parallel ((x, y) \cdot Q)))$
by (*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)
with $\langle x \# P \rangle \langle y \# P \rangle$ **have** $(\nu y)(P \parallel ((x, y) \cdot Q)) = (\nu x)(P \parallel Q)$
by (*simp add: eqvts*)
moreover from $\langle x \# P' \rangle \langle y \# P' \rangle \langle y \# Q' \rangle \langle xvec \#* x \rangle \langle y \# xvec \rangle$ **have** $(\nu y)((\nu * xvec)(P' \parallel ((x, y) \cdot Q))) = (\nu x)((\nu * xvec)(P' \parallel Q'))$
by (*subst alphaRes[of y]*) (*auto simp add: resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto \tau \prec (\nu x)((\nu * xvec)(P' \parallel Q'))$
by *simp*
moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle \langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu x)((\nu * xvec)(P' \parallel Q')), (\nu * xvec)(P' \parallel (\nu x)Q')) \in Rel$
by (*rule C2*)
ultimately show *?case by blast*
qed
next
case (*cBrMerge* $\Psi_Q M N P' A_P \Psi_P xQ' A_Q$)
have $QTrans: \Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto \imath M(\imath N) \prec xQ'$ **and** $FrQ: extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle$ **by** *fact+*
have $PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\imath N) \prec P'$ **and** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
from $\langle x \# \alpha \rangle \langle \alpha = \imath M(\imath N) \rangle$ **have** $x \# M$ **and** $x \# N$ **by** *simp+*
from $\langle x \# P' \parallel xQ' \rangle$ **have** $x \# xQ'$ **by** *simp*
from $FrP \langle A_P \#* x \rangle \langle x \# P \rangle$ **have** $x \# \Psi_P$
by (*drule-tac extractFrameFresh*) *auto*
from $PTrans \langle x \# P \rangle \langle x \# N \rangle$ **have** $x \# P'$
by (*rule brinputFreshDerivative*)
have $x \# (\nu x)Q$ **by** (*simp add: abs-fresh*)
with $FrQ \langle A_Q \#* x \rangle$ **have** $x \# \Psi_Q$
by (*drule-tac extractFrameFresh*) *auto*
from $\langle A_P \#* (\nu x)Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* Q$ **by** *simp*
from $\langle A_Q \#* (\nu x)Q \rangle \langle A_Q \#* x \rangle$ **have** $A_Q \#* Q$ **by** *simp*
from $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$
have $x \# (\Psi \otimes \Psi_P)$ **by** *force*
from $\langle \Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto \imath M(\imath N) \prec xQ' \rangle \langle x \# (\Psi \otimes \Psi_P) \rangle \langle x \# M \rangle \langle x \# N \rangle \langle x \# xQ' \rangle$
show *?case*
proof (*induct rule: resBrInputCases*)
case (*cRes* Q')
have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\imath N) \prec Q'$ **by** *fact*
with $\langle A_Q \#* Q \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* Q'$
by (*elim brinputFreshChainDerivative*)
with $\langle extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $FrQ: (\nu x)(extractFrame Q) = \langle A_Q, \Psi_Q \rangle$ **by** *simp*

then obtain $y \# A_{Q'}$ **where** $A: A_Q = y \# A_{Q'}$ **by** (*cases* A_Q) *auto*
with $\langle A_Q \# \Psi \rangle \langle A_Q \# P \rangle \langle A_Q \# P' \rangle \langle A_Q \# \Psi_P \rangle \langle A_Q \# Q \rangle \langle A_Q \# M \rangle$
 $\langle A_Q \# Q' \rangle \langle A_Q \# N \rangle$
have $A_{Q'} \# \Psi$ **and** $A_{Q'} \# P$ **and** $A_{Q'} \# \Psi_P$ **and** $A_{Q'} \# Q$ **and** $A_Q \# Q'$
and $y \# \Psi$ **and** $y \# P$ **and** $y \# P'$ **and** $y \# \Psi_P$ **and** $y \# Q$ **and** $y \# M$ **and**
 $y \# Q'$ **and** $y \# N$
and $A_{Q'} \# M$
by (*simp*) +
from $A \langle A_P \# A_Q \rangle$ **have** $A_P \# A_{Q'}$ **and** $y \# A_P$ **by** (*simp add: fresh-star-list-cons*) +
from $A \langle A_Q \# x \rangle$ **have** $x \neq y$ **and** $x \# A_{Q'}$ **by** (*simp add: fresh-list-cons*) +

with $A \langle \text{distinct } A_Q \rangle$ **have** $y \# A_{Q'}$
by (*induct* $A_{Q'}$) (*auto simp add: fresh-list-nil fresh-list-cons*)

note $PTrans FrP$
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_{iM} (N) \prec Q' \rangle$ **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \triangleright ([(x, y)] \cdot Q) \mapsto_{iM} (N) \prec Q'$
by (*rule semantics.eqvt*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# \Psi_P \rangle \langle y \# \Psi_P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# N \rangle \langle y \# N \rangle$
have $\Psi \otimes \Psi_P \triangleright ([(x, y)] \cdot Q) \mapsto_{iM} (N) \prec ([(x, y)] \cdot Q')$
by (*simp add: eqvts*)
moreover from $A \langle A_Q \# x \rangle FrxQ$ **have** $FrQ: \text{extractFrame}([(x, y)] \cdot Q)$
 $= \langle A_{Q'}, \Psi_Q \rangle$ **and** $y \# \text{extractFrame } Q$
by (*clarsimp simp add: alpha' eqvts frame.inject fresh-list-cons name-swap*) +
moreover from $\langle A_P \# Q \rangle$ **have** $([(x, y)] \cdot A_P) \# ([(x, y)] \cdot Q)$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \# x \rangle \langle y \# A_P \rangle$ **have** $A_P \# ([(x, y)] \cdot Q)$ **by** *simp*
moreover from $\langle A_{Q'} \# Q \rangle$ **have** $([(x, y)] \cdot A_{Q'}) \# ([(x, y)] \cdot Q)$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle x \# A_{Q'} \rangle \langle y \# A_{Q'} \rangle$ **have** $A_{Q'} \# ([(x, y)] \cdot Q)$ **by** *simp*
ultimately have $\Psi \triangleright (P \parallel ([(x, y)] \cdot Q)) \mapsto_{iM} (N) \prec (P' \parallel ([(x, y)] \cdot Q'))$

using $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle \langle A_P \# A_{Q'} \rangle \langle A_{Q'} \# \Psi \rangle \langle A_{Q'} \# P \rangle \langle A_P \# M \rangle \langle A_{Q'} \# M \rangle$
by (*intro BrMerge*)
with $\langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# N \rangle$ **have** $\Psi \triangleright (\nu y)(P \parallel ([(x, y)] \cdot Q)) \mapsto_{iM} (N) \prec (\nu y)(P' \parallel ([(x, y)] \cdot Q'))$
by (*intro Scope auto*)
moreover from $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$ **have** $(\nu y)(P \parallel ([(x, y)] \cdot Q)) = (\nu x)(([(x, y)] \cdot (P \parallel ([(x, y)] \cdot Q)))$
by (*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)
moreover with $\langle x \# P \rangle \langle y \# P \rangle$ **have** $(\nu y)(P \parallel ([(x, y)] \cdot Q)) = (\nu x)(P \parallel Q)$

by (*simp add: eqvts*)
moreover from $\langle x \# P' \rangle \langle y \# P' \rangle \langle y \# Q' \rangle$ **have** $(\nu y)(P' \parallel ([(x, y)] \cdot Q')) = (\nu x)(P' \parallel Q')$

by (*subst alphaRes[of y]*) (*auto simp add: resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\text{finTrans}: \Psi \triangleright (\nu x)(P \parallel Q) \mapsto_{iM} (N) \prec (\nu x)(P' \parallel Q')$

by *simp*
 from $\langle x \# \Psi \rangle \langle x \# P' \rangle$ have $[x] \#* P'$ and $[x] \#* \Psi$ by *simp+*
 then have $(\Psi, (\nu*[x])(P' \parallel Q'), (P' \parallel ((\nu*[x])Q')) \in Rel$
 by *(rule C4)*
 then have $(\Psi, (\nu x)(P' \parallel Q'), (P' \parallel (\nu x)Q')) \in Rel$
 by *simp*
 with *finTrans* show ?case by *blast*
 qed
 next
 case *(cBrComm1 $\Psi_Q M N P' A_P \Psi_P xvec xQ' A_Q$)*
 have *QTrans*: $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto_{iM} (\nu*xvec)\langle N \rangle \prec xQ'$ and *FrQ*: *extractFrame* $(\nu x)Q = \langle A_Q, \Psi_Q \rangle$ by *fact+*
 have *PTrans*: $\Psi \otimes \Psi_Q \triangleright P \mapsto_{iM} \langle N \rangle \prec P'$ and *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ by *fact+*
 from $\langle bn \alpha \#* x \rangle \langle_{iM} (\nu*xvec)\langle N \rangle = \alpha \rangle$ have $x \# xvec$ by *force*
 have $x \# (\nu x)Q$ by *(simp add: abs-fresh)*
 with *QTrans* have $x \# N$ and $x \# xQ'$ using $\langle x \# xvec \rangle \langle xvec \#* M \rangle \langle distinct\ xvec \rangle$
 by *(force intro: brouputFreshDerivative)+*
 from *PTrans* $\langle x \# P \rangle \langle x \# N \rangle$ have $x \# P'$ by *(rule brinputFreshDerivative)*
 from $\langle x \# (\nu x)Q \rangle$ *FrQ* $\langle A_Q \#* x \rangle$ have $x \# \Psi_Q$
 by *(drule-tac extractFrameFresh) auto*
 from $\langle x \# P \rangle$ *FrP* $\langle A_P \#* x \rangle$ have $x \# \Psi_P$
 by *(drule-tac extractFrameFresh) auto*
 from $\langle A_P \#* (\nu x)Q \rangle \langle A_P \#* x \rangle$ have $A_P \#* Q$ by *simp*
 from $\langle A_Q \#* (\nu x)Q \rangle \langle A_Q \#* x \rangle$ have $A_Q \#* Q$ by *simp*
 note *QTrans*
 moreover from $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$ have $x \# \Psi \otimes \Psi_P$ by *force*
 moreover from $\langle x \# xvec \rangle$ have $xvec \#* x$ by *simp*
 from $\langle x \# \alpha \rangle \langle_{iM} (\nu*xvec)\langle N \rangle = \alpha \rangle$ have $x \#_{iM} (\nu*xvec)\langle N \rangle$ by *simp*
 moreover note $\langle x \# xQ' \rangle$

 moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ have $xvec \#* (\Psi \otimes \Psi_P)$ by *force*
 moreover from $\langle xvec \#* (\nu x)Q \rangle \langle x \# xvec \rangle$ have $xvec \#* Q$ by *simp*
 moreover note $\langle xvec \#* M \rangle$

 moreover from $\langle xvec \#* \Psi \rangle \langle xvec \#* \Psi_P \rangle$ have $bn(iM(\nu*xvec)\langle N \rangle) \#* (\Psi \otimes \Psi_P)$ by *force*
 moreover from $\langle xvec \#* (\nu x)Q \rangle \langle x \# xvec \rangle$ have $bn(iM(\nu*xvec)\langle N \rangle) \#* Q$
 by *simp*
 moreover from $\langle xvec \#* P \rangle$ have $bn(iM(\nu*xvec)\langle N \rangle) \#* P$ by *simp*
 from $\langle xvec \#* \Psi \rangle$ have $bn(iM(\nu*xvec)\langle N \rangle) \#* \Psi$ by *simp*
 from $\langle A_Q \#* xvec \rangle \langle A_Q \#* M \rangle \langle A_Q \#* N \rangle$ have $A_Q \#* (iM(\nu*xvec)\langle N \rangle)$ by *simp*
 have *object* $(iM(\nu*xvec)\langle N \rangle) = Some N$ by *simp*
 have $bn(iM(\nu*xvec)\langle N \rangle) = xvec$ by *simp*
 have *subject* $(iM(\nu*xvec)\langle N \rangle) = Some M$ by *simp*
 from $\langle xvec \#* M \rangle$ have $bn(iM(\nu*xvec)\langle N \rangle) \#* subject(iM(\nu*xvec)\langle N \rangle)$ by *simp*

ultimately show *?case*
using $\langle x \# \text{iM}(\nu * \text{xvec}) \langle N \rangle \rangle \langle \text{bn}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) \#* P \rangle \langle \text{bn}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) \#* \Psi \rangle$
 $\langle \text{object}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) = \text{Some } N \rangle$
 $\langle \text{bn}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) = \text{xvec} \rangle \langle \text{subject}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) = \text{Some } M \rangle \langle A_Q \#* (\text{iM}(\nu * \text{xvec}) \langle N \rangle) \rangle$
proof(*induct rule: resBrOutputCases'*)
case(*cBrOpen* M'' xvec1 xvec2 y N' Q')
then have $\text{Eq}: \text{iM}(\nu * \text{xvec}) \langle N \rangle = \text{iM}''(\nu * (\text{xvec1} @ y \# \text{xvec2})) \langle N' \rangle$ **by**
simp
from $\langle x \# \text{iM}(\nu * \text{xvec}) \langle N \rangle \rangle \text{Eq}$ **have** $x \# \text{xvec1}$ **and** $x \neq y$ **and** $x \# \text{xvec2}$
and $x \# M''$
by *simp+*
from $\langle \text{bn}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) \#* P \rangle \text{Eq}$ **have** $(\text{xvec1} @ \text{xvec2}) \#* P$ **and** $y \# P$
by *simp+*
from $\langle A_Q \#* (\text{iM}(\nu * \text{xvec}) \langle N \rangle) \rangle \text{Eq}$ **have** $(\text{xvec1} @ \text{xvec2}) \#* A_Q$ **and** $y \# A_Q$
and $A_Q \#* M''$ **by** *simp+*
from $\langle \text{bn}(\text{iM}(\nu * \text{xvec}) \langle N \rangle) \#* \Psi \rangle \text{Eq}$ **have** $(\text{xvec1} @ \text{xvec2}) \#* \Psi$ **and** $y \# \Psi$
by *simp+*
from Eq **have** $N = N'$ **and** $\text{xvec} = \text{xvec1} @ y \# \text{xvec2}$ **and** $M = M''$ **by**(*simp*
add: action.inject)
from $\langle x \# P \rangle \langle y \# P \rangle \langle x \neq y \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto \text{iM} \langle N \rangle \prec P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto \text{iM}(\langle [(y, x)] \cdot N \rangle) \prec \langle [(y, x)] \cdot P' \rangle$
by(*intro brinputAlpha*[**where** $\text{xvec} = [y]$]) (*auto simp add: calc-atm*)
then have $P\text{Trans}: \Psi \otimes \Psi_Q \triangleright P \mapsto \text{iM}(\langle [(x, y)] \cdot N \rangle) \prec \langle [(x, y)] \cdot P' \rangle$
by(*simp add: name-swap*)

from $\langle \Psi \otimes \Psi_P \triangleright [(x, y)] \cdot Q \mapsto \text{iM}''(\nu * (\text{xvec1} @ \text{xvec2})) \langle N' \rangle \prec Q' \rangle$
have $[(x, y)] \cdot (\Psi \otimes \Psi_P \triangleright [(x, y)] \cdot Q \mapsto \text{iM}''(\nu * (\text{xvec1} @ \text{xvec2})) \langle N' \rangle) \prec Q'$
by *simp*
then have $[(x, y)] \cdot (\Psi \otimes \Psi_P) \triangleright \langle [(x, y)] \cdot \langle [(x, y)] \cdot Q \rangle \rangle \mapsto \text{i}(\langle [(x, y)] \cdot M'' \rangle(\nu * (\langle [(x, y)] \cdot (\text{xvec1} @ \text{xvec2}) \rangle) \langle [(x, y)] \cdot N' \rangle) \prec \langle [(x, y)] \cdot Q' \rangle$
by(*simp add: eqvts*)
with $\langle x \# (\Psi \otimes \Psi_P) \rangle \langle y \# (\Psi \otimes \Psi_P) \rangle \langle x \# \text{xvec1} \rangle \langle y \# \text{xvec1} \rangle \langle x \# \text{xvec2} \rangle$
 $\langle y \# \text{xvec2} \rangle \langle x \# M'' \rangle \langle y \# M'' \rangle$
have $Q\text{Trans}: \Psi \otimes \Psi_P \triangleright Q \mapsto \text{iM}''(\nu * (\text{xvec1} @ \text{xvec2})) \langle [(x, y)] \cdot N' \rangle$
 $\prec \langle [(x, y)] \cdot Q' \rangle$
by *simp*
with $\langle A_Q \#* x \rangle \langle y \# A_Q \rangle \langle \text{distinct } \text{xvec1} \rangle \langle \text{distinct } \text{xvec2} \rangle \langle \text{xvec1} \#* \text{xvec2} \rangle$
 $\langle \text{xvec1} \#* M'' \rangle \langle \text{xvec2} \#* M'' \rangle$
 $\langle (\text{xvec1} @ \text{xvec2}) \#* A_Q \rangle$
have $A_Q \#* \langle [(x, y)] \cdot Q' \rangle$ **using** $\langle A_Q \#* Q \rangle$
by(*elim broutputFreshChainDerivative(2)*) (*assumption | simp*)

from $\langle \text{extractFrame}(\langle \nu x \rangle Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $\text{Fr}xQ: \langle \nu x \rangle(\text{extractFrame } Q) = \langle A_Q, \Psi_Q \rangle$ **by** *simp*
then obtain $z A_Q'$ **where** $A: A_Q = z \# A_Q'$ **by**(*cases* A_Q) *auto*
with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle (\text{xvec1} @ \text{xvec2}) \#* A_Q \rangle$
 $\langle A_Q \#* M'' \rangle \langle A_Q \#* \langle [(x, y)] \cdot Q' \rangle \rangle \langle y \# A_Q \rangle \langle A_Q \#* N \rangle$

have $A_{Q'} \# \Psi$ **and** $A_{Q'} \# P$ **and** $A_{Q'} \# \Psi_P$ **and** $A_{Q'} \# Q$
and $z \# \Psi$ **and** $z \# P$ **and** $z \# P'$ **and** $z \# \Psi_P$ **and** $z \# Q$ **and** $z \# xvec1$
and $z \# xvec2$
and $z \# M''$ **and** $z \# ((x, y) \cdot Q')$ **and** $A_{Q'} \# M''$ **and** $z \neq y$ **and** $z \#$
 $(xvec1 @ xvec2)$
by *auto*
from $A \langle A_P \# A_Q \rangle$ **have** $A_P \# A_{Q'}$ **and** $z \# A_P$ **by** *simp+*
from $A \langle A_Q \# x \rangle$ **have** $x \neq z$ **and** $x \# A_{Q'}$ **by** *simp+*

from $\langle A_Q \# (iM(\nu * xvec) \langle N \rangle) \rangle A$ **have** $z \# M$ **and** $z \# xvec$ **and** $z \# N$ **by**
simp+

from $\langle y \in \text{supp } N \rangle \langle N = N' \rangle$ **have** $y \in \text{supp } N$ **by** *simp*
then **have** $x \in \text{supp } ((x, y) \cdot N)$
by *(rule swap-supp)*
then **have** $z \text{supp}: z \in \text{supp } ((x, z) \cdot [(x, y) \cdot N])$
by *(rule swap-supp')*

from $\langle \text{distinct } A_Q \rangle A$ **have** $z \# A_{Q'}$
by *(induct A_{Q'}) (auto simp add: fresh-list-nil fresh-list-cons)*
from $PTrans \langle x \# P \rangle \langle z \# P \rangle \langle x \neq z \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\langle [(x, z)] \cdot$
 $\langle [(x, y) \cdot N] \rangle \prec \langle [(x, z)] \cdot [(x, y) \cdot P'] \rangle)$
by *(elim brinputAlpha[where xvec=[x]] (auto simp add: calc-atm))*
moreover **note** *FrP*
moreover **from** $QTrans$ **have** $([(x, z)] \cdot (\Psi \otimes \Psi_P)) \triangleright ((x, z) \cdot Q) \mapsto ((x,$
 $z) \cdot (iM''(\nu * (xvec1 @ xvec2)) \langle [(x, y) \cdot N'] \rangle \prec ((x, y) \cdot Q')))$
by *(rule semantics.eqt)*
with $\langle x \# \Psi \rangle \langle z \# \Psi \rangle \langle x \# \Psi_P \rangle \langle z \# \Psi_P \rangle \langle x \# M'' \rangle \langle z \# M'' \rangle \langle x \# xvec1 \rangle \langle x \#$
 $xvec2 \rangle \langle z \# xvec1 \rangle \langle z \# xvec2 \rangle$
have $\Psi \otimes \Psi_P \triangleright ((x, z) \cdot Q) \mapsto_i M''(\nu * (xvec1 @ xvec2)) \langle ((x, z) \cdot [(x, y)$
 $\cdot N'] \rangle \prec ((x, z) \cdot [(x, y) \cdot Q'])$
by *(simp add: eqts)*
moreover **from** $A \langle A_Q \# x \rangle FrxQ$ **have** $\text{extractFrame}(\langle [(x, z)] \cdot Q \rangle) = \langle A_{Q'},$
 $\Psi_Q \rangle$
by *(clarsimp simp add: alpha' eqts frame.inject fresh-list-cons name-swap)*
moreover **from** $\langle A_P \# Q \rangle$ **have** $([(x, z)] \cdot A_P) \# ((x, z) \cdot Q)$
by *(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])*
with $\langle A_P \# x \rangle \langle z \# A_P \rangle$ **have** $A_P \# ((x, z) \cdot Q)$ **by** *simp*
moreover **from** $\langle A_{Q'} \# Q \rangle$ **have** $([(x, z)] \cdot A_{Q'}) \# ((x, z) \cdot Q)$
by *(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])*
with $\langle x \# A_{Q'} \rangle \langle z \# A_{Q'} \rangle$ **have** $A_{Q'} \# ((x, z) \cdot Q)$ **by** *simp*
ultimately **have** $\Psi \triangleright (P \parallel ((x, z) \cdot Q)) \mapsto_i M(\nu * (xvec1 @ xvec2)) \langle ((x,$
 $z) \cdot [(x, y) \cdot N] \rangle \prec (([(x, z)] \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q']))$
using $\langle M = M'' \rangle \langle N = N' \rangle \langle A_P \# \Psi \rangle \langle A_P \# P \rangle \langle A_P \# A_{Q'} \rangle \langle A_{Q'} \# \Psi \rangle$
 $\langle A_{Q'} \# P \rangle \langle (xvec1 @ xvec2) \# P \rangle \langle A_P \# M \rangle \langle A_{Q'} \# M'' \rangle$
by *(intro BrComm1 (assumption | simp) +)*
then **have** $\text{permTrans}: \Psi \triangleright (\nu z)(P \parallel ((x, z) \cdot Q)) \mapsto_i M(\nu * (xvec1 @ z \# xvec2)) \langle ((x,$
 $z) \cdot [(x, y) \cdot N] \rangle \prec (([(x, z)] \cdot [(x, y) \cdot P'] \parallel ((x, z) \cdot [(x, y) \cdot Q']))$
using $z \text{supp} \langle z \# \Psi \rangle \langle z \# M \rangle \langle z \# xvec1 \rangle \langle z \# xvec2 \rangle$

by(*rule BrOpen*)

from $\langle x \# P \rangle \langle z \# P \rangle \langle z \# Q \rangle$ **have** $(\nu z)(P \parallel ((x, z) \cdot Q)) = (\nu x)((x, z) \cdot (P \parallel ((x, z) \cdot Q)))$

by(*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)

with $\langle x \# P \rangle \langle z \# P \rangle$ **have** $(\nu z)(P \parallel ((x, z) \cdot Q)) = (\nu x)(P \parallel Q)$

by(*simp add: eqvts*)

with permTrans **have** *permTrans2*: $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_{iM} (\nu*(xvec1@z\#xvec2))(\langle (x, z) \rangle \cdot [(x, y) \cdot N]) \prec (\langle (x, z) \rangle \cdot [(x, y) \cdot P'] \parallel \langle (x, z) \rangle \cdot [(x, y) \cdot Q'])$

by *simp*

then **have** $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_{RBrOut M} (\nu*(xvec1@z\#xvec2))(\langle (x, z) \rangle \cdot [(x, y) \cdot N]) \prec' (\langle (x, z) \rangle \cdot [(x, y) \cdot P'] \parallel \langle (x, z) \rangle \cdot [(x, y) \cdot Q'])$

by(*simp add: residualInject*)

then **have** *permTrans3*: $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_{RBrOut M} (\nu*(xvec1@z\#xvec2))(\langle (x, z) \rangle \cdot [(x, y) \cdot N]) \prec' (\langle (x, z) \rangle \cdot [(x, y) \cdot (P' \parallel Q')])$

by *simp*

from $\langle z \# N \rangle \langle x \neq z \rangle \langle z \neq y \rangle$

have $z \# ((x, y) \cdot N)$

by (*metis calc-atm(2) calc-atm(3) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def*)

from $\langle z \# P' \rangle \langle x \neq z \rangle \langle z \neq y \rangle$

have $z \# ((x, y) \cdot P')$

by (*metis calc-atm(2) calc-atm(3) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def*)

from $\langle z \# ((x, y) \cdot N) \rangle$ **have** $x \# ((x, z) \cdot [(x, y) \cdot N])$

by (*metis calc-atm(2) calc-atm(3) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def*)

from $\langle z \# ((x, y) \cdot P') \rangle$ **have** $x \# ((x, z) \cdot [(x, y) \cdot P'])$

by (*metis calc-atm(2) calc-atm(3) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def*)

moreover **from** $\langle z \# [(x, y) \cdot Q'] \rangle$

have $x \# ((x, z) \cdot [(x, y) \cdot Q'])$

by (*metis calc-atm(2) calc-atm(3) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rev-conv swap-name-def*)

ultimately **have** $x \# ((x, z) \cdot [(x, y) \cdot (P' \parallel Q')])$

by *simp*

have $z \in \text{set}((xvec1@z\#xvec2))$

by *simp*

from $\langle x \# ((x, z) \cdot [(x, y) \cdot N]) \rangle \langle x \# ((x, z) \cdot [(x, y) \cdot (P' \parallel Q')]) \rangle \langle z \in \text{set}((xvec1@z\#xvec2)) \rangle$

have *eq1*: $(\nu*(xvec1@z\#xvec2))(\langle (x, z) \rangle \cdot [(x, y) \cdot N]) \prec' (\langle (x, z) \rangle \cdot [(x, y) \cdot (P' \parallel Q')]) = (\nu*((z, x) \cdot (xvec1@z\#xvec2))(\langle (z, x) \rangle \cdot [(z, x) \cdot [(x, z) \cdot [(x, y) \cdot N]] \prec' ((z, x) \cdot [(x, z) \cdot [(x, y) \cdot (P' \parallel Q')])])$

by(*rule boundOutputChainSwap*)

have *eq2*: $(\nu*((z, x) \cdot (xvec1@z\#xvec2))(\langle (z, x) \rangle \cdot [(z, x) \cdot [(x, z) \cdot [(x, y) \cdot N]] \prec' ((z, x) \cdot [(x, z) \cdot [(x, y) \cdot (P' \parallel Q')])]) = (\nu*((z, x) \cdot (xvec1@z\#xvec2))(\langle (z, x) \rangle \cdot [(z, x) \cdot [(x, y) \cdot (P' \parallel Q')]) \prec' ((z, x) \cdot [(x, y) \cdot (P' \parallel Q')])$

by *auto* (*metis perm-swap(2)*)

from $\langle x \# xvec1 \rangle \langle x \# xvec2 \rangle \langle z \# xvec1 \rangle \langle z \# xvec2 \rangle$

have $([(z, x)] \cdot (xvec1 @ z \# xvec2)) = (xvec1 @ x \# xvec2)$
by *(simp add: eqvts swap-simps)*
then have $eq3: (\nu * ([(z, x)] \cdot (xvec1 @ z \# xvec2))) \Downarrow (([(x, y)] \cdot N) \prec' (([(x, y)] \cdot (P' \parallel Q')))) = (\nu * (xvec1 @ x \# xvec2)) \Downarrow (([(x, y)] \cdot N) \prec' (([(x, y)] \cdot (P' \parallel Q'))))$
by *simp*

from *permTrans3 eq1 eq2 eq3* **have** $noXZTrans: \Psi \triangleright (\nu x)(P \parallel Q) \mapsto RBrOut M (\nu * (xvec1 @ x \# xvec2)) \Downarrow (([(x, y)] \cdot N) \prec' (([(x, y)] \cdot (P' \parallel Q'))))$
by *simp*

from $\langle x \# N \rangle$
have $y \# (([(x, y)] \cdot N)$
by *(metis calc-atm(2) fresh-right name-prm-name.simps(2) name-prm-name-def singleton-rew-conv swap-name-def swap-simps(2))*
moreover from $\langle x \# P' \rangle \langle x \# Q' \rangle$
have $y \# (([(x, y)] \cdot (P' \parallel Q'))$
by *(metis fresh-left psi.fresh(5) singleton-rew-conv swap-simps(2))*
moreover have $x \in set (xvec1 @ x \# xvec2)$
by *simp*
ultimately have $eq1: (\nu * (xvec1 @ x \# xvec2)) \Downarrow (([(x, y)] \cdot N) \prec' (([(x, y)] \cdot (P' \parallel Q')))) = (\nu * ([[(x, y)] \cdot (xvec1 @ x \# xvec2)]) \Downarrow ([[(x, y)] \cdot [(x, y)] \cdot N] \prec' ([[(x, y)] \cdot [(x, y)] \cdot (P' \parallel Q')]))$
by *(rule boundOutputChainSwap)*
have $eq2: (\nu * ([[(x, y)] \cdot (xvec1 @ x \# xvec2)]) \Downarrow ([[(x, y)] \cdot [(x, y)] \cdot N] \prec' ([[(x, y)] \cdot [(x, y)] \cdot (P' \parallel Q')])) = (\nu * ([[(x, y)] \cdot (xvec1 @ x \# xvec2)]) \Downarrow N \prec' (P' \parallel Q'))$
by *simp*
from $\langle x \# xvec1 \rangle \langle x \# xvec2 \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$
have $eq3: ([[(x, y)] \cdot (xvec1 @ x \# xvec2)]) = (xvec1 @ y \# xvec2)$
by *(simp add: eqvts swap-simps)*

from *eq1 eq2 eq3 noXZTrans* **have** $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto RBrOut M ((\nu * (xvec1 @ y \# xvec2)) \Downarrow N \prec' (P' \parallel Q'))$
by *simp*
then have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_i M ((\nu * (xvec1 @ y \# xvec2)) \Downarrow \langle N \rangle \prec (P' \parallel Q'))$
by *(simp add: residualInject)*

with $\langle xvec = xvec1 @ y \# xvec2 \rangle$
have $\Psi \triangleright (\nu x)(P \parallel Q) \mapsto_i M ((\nu * xvec) \Downarrow \langle N \rangle \prec (P' \parallel Q'))$
by *simp*
moreover have $(\Psi, P' \parallel Q', P' \parallel Q') \in Rel$
by *(rule C1)*

ultimately show *?case*
by *force*

next
case *(cRes Q')*
from $\langle x \# iM((\nu * xvec) \Downarrow \langle N \rangle) \rangle$
have $x \# M$ **and** $x \# xvec$ **and** $x \# N$
by *simp+*

have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec)\langle N \rangle \prec Q'$ **by fact**
with $\langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle \langle xvec \#* M \rangle \langle distinct\ xvec \rangle$ **have** $A_Q \#* Q'$
by(*force dest: brotputFreshChainDerivative*)

with $\langle extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $FrQ: (\nu x)(extractFrame$
 $Q) = \langle A_Q, \Psi_Q \rangle$ **by simp**
then obtain $y A_Q'$ **where** $A: A_Q = y \# A_Q'$ **by**(*cases A_Q*) **auto**
with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle$
 $\langle A_Q \#* M \rangle \langle A_Q \#* Q' \rangle \langle A_Q \#* N \rangle$
have $A_Q' \#* \Psi$ **and** $A_Q' \#* P$ **and** $A_Q' \#* \Psi_P$ **and** $A_Q' \#* Q$ **and** $A_Q \#*$
 $xvec$ **and** $A_Q \#* Q'$
and $y \# \Psi$ **and** $y \# P$ **and** $y \# P'$ **and** $y \# \Psi_P$ **and** $y \# Q$ **and** $y \# xvec$
and $y \# M$ **and** $y \# Q'$
and $A_Q' \#* M$ **and** $y \# N$
by(*simp*)**+**
from $A \langle A_P \#* A_Q \rangle$ **have** $A_P \#* A_Q'$ **and** $y \# A_P$ **by**(*simp add: fresh-star-list-cons*)**+**
from $A \langle A_Q \#* x \rangle$ **have** $x \neq y$ **and** $x \# A_Q'$ **by**(*simp add: fresh-list-cons*)**+**

with $A \langle distinct\ A_Q \rangle$ **have** $y \# A_Q'$
by(*induct A_Q'*) (*auto simp add: fresh-list-nil fresh-list-cons*)

from $\langle x \# N \rangle$ **have** $y \# [(x, y)] \cdot N$
by (*metis calc-atm(2) fresh-right name-prm-name.simps(2) name-prm-name-def*
singleton-rew-conv swap-name-def swap-simps(2))

from $\langle x \# P \rangle \langle y \# P \rangle \langle x \neq y \rangle \langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M\langle N \rangle \prec P' \rangle$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\langle [(y, x)] \cdot N \rangle) \prec [(y, x)] \cdot P'$
by(*elim brinputAlpha[where xvec=[y]]*) (*auto simp add: calc-atm*)
then have $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\langle [(x, y)] \cdot N \rangle) \prec [(x, y)] \cdot P'$
by(*simp add: name-swap*)
moreover note FrP
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu * xvec)\langle N \rangle \prec Q' \rangle$ **have** $([(x, y)] \cdot$
 $(\Psi \otimes \Psi_P)) \triangleright (([(x, y)] \cdot Q) \mapsto_i M(\nu * xvec)\langle N \rangle \prec Q')$
by(*rule semantics.eqvt*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# \Psi_P \rangle \langle y \# \Psi_P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# xvec \rangle \langle y \#$
 $xvec \rangle$
have $\Psi \otimes \Psi_P \triangleright (([(x, y)] \cdot Q) \mapsto_i M(\nu * xvec)\langle [(x, y)] \cdot N \rangle) \prec (([(x, y)] \cdot$
 $Q')$
by(*simp add: eqvts*)
moreover from $A \langle A_Q \#* x \rangle FrQ$ **have** $FrQ: extractFrame([(x, y)] \cdot Q)$
 $= \langle A_Q', \Psi_Q \rangle$
by(*clarsimp simp add: alpha' eqvts frame.inject fresh-list-cons name-swap*)
moreover from $\langle A_P \#* Q \rangle$ **have** $([(x, y)] \cdot A_P) \#* (([(x, y)] \cdot Q)$ **by**(*simp*
add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with $\langle A_P \#* x \rangle \langle y \# A_P \rangle$ **have** $A_P \#* (([(x, y)] \cdot Q)$ **by simp**
moreover from $\langle A_Q' \#* Q \rangle$ **have** $([(x, y)] \cdot A_Q') \#* (([(x, y)] \cdot Q)$ **by**(*simp*
add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with $\langle x \# A_Q' \rangle \langle y \# A_Q' \rangle$ **have** $A_Q' \#* (([(x, y)] \cdot Q)$ **by simp**
ultimately have $\Psi \triangleright (P \parallel (([(x, y)] \cdot Q)) \mapsto_i M(\nu * xvec)\langle [(x, y)] \cdot N \rangle) \prec$

```

((x, y) · P') || ((x, y) · Q')
  using ⟨AP #* Ψ⟩ ⟨AP #* P⟩ ⟨AP #* AQ'⟩ ⟨AQ' #* Ψ⟩ ⟨AQ' #* P⟩ ⟨xvec #*
P⟩ ⟨AP #* M⟩ ⟨AQ' #* M⟩
  by(intro BrComm1) (assumption | simp)+
  with ⟨y # Ψ⟩ ⟨y # M⟩ ⟨y # xvec⟩ ⟨y # [(x, y) · N]⟩ have Ψ ▷ (νy)(P || ((x,
y) · Q)) ⟶iM(ν*xvec)⟨[(x, y) · N]⟩ < (νy)⟨[(x, y) · P'] || ((x, y) · Q')⟩
  by(intro Scope) auto
  moreover from ⟨x # P⟩ ⟨y # P⟩ ⟨y # Q⟩ have (νy)(P || ((x, y) · Q)) =
(νx)⟨[(x, y) · (P || ((x, y) · Q))⟩
  by(subst alphaRes[of x]) (auto simp add: calc-atm fresh-left name-swap)
  with ⟨x # P⟩ ⟨y # P⟩ have (νy)(P || ((x, y) · Q)) = (νx)(P || Q)
  by(simp add: eqvts)
  moreover from ⟨y # P'⟩ ⟨y # Q'⟩ ⟨x # xvec⟩ ⟨y # xvec⟩ have (νy)⟨[(x, y)
· P'] || ((x, y) · Q')⟩ = (νx)⟨P' || Q'⟩
  by(subst alphaRes[of y]) (auto simp add: resChainFresh calc-atm eqvts
fresh-left name-swap)
  ultimately have Ψ ▷ (νx)(P || Q) ⟶iM(ν*xvec)⟨[(x, y) · N]⟩ < (νx)⟨P'
|| Q'⟩
    by simp
    with ⟨x # N⟩ ⟨y # N⟩
    have Trans: Ψ ▷ (νx)(P || Q) ⟶iM(ν*xvec)⟨N⟩ < (νx)⟨P' || Q'⟩
    by simp
    from ⟨x # P'⟩ ⟨x # Ψ⟩
    have (Ψ, (ν*[x])⟨P' || Q'⟩, (P' || ((ν*[x])Q'))) ∈ Rel
    by(intro C4) simp+
    then have Relation: (Ψ, (νx)⟨P' || Q'⟩, (P' || ((νx)Q'))) ∈ Rel
    by simp
    from Trans Relation show ?case
    by blast
qed
next
case(cBrComm2 ΨQ M xvec N P' AP ΨP xQ' AQ)
from ⟨x # α⟩ ⟨iM(ν*xvec)⟨N⟩ = α⟩
have x # M x # xvec x # N
  by force+
  have QTrans: Ψ ⊗ ΨP ▷ (νx)Q ⟶iM⟨N⟩ < xQ' and FrQ: extract-
Frame((νx)Q) = ⟨AQ, ΨQ⟩ by fact+
  have PTrans: Ψ ⊗ ΨQ ▷ P ⟶iM(ν*xvec)⟨N⟩ < P' and FrP: extractFrame
P = ⟨AP, ΨP⟩ by fact+
  from PTrans ⟨x # P⟩ have x # N and x # P' using ⟨x # xvec⟩ ⟨xvec #* M⟩
⟨distinct xvec⟩
  by(force intro: broutputFreshDerivative)+
  have x # (νx)Q by(simp add: abs-fresh)
  with FrQ ⟨AQ #* x⟩ have x # ΨQ
  by(drule-tac extractFrameFresh) auto
  from ⟨x # P⟩ FrP ⟨AP #* x⟩ have x # ΨP
  by(drule-tac extractFrameFresh) auto
  from ⟨AP #* (νx)Q⟩ ⟨AP #* x⟩ have AP #* Q by simp
  from ⟨AQ #* (νx)Q⟩ ⟨AQ #* x⟩ have AQ #* Q by simp

```


from $\langle x \# xvec \rangle \langle xvec \#* (\nu x)Q \rangle$ **have** $xvec \#* Q$ **by** *simp*

from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P' \rangle$ **have** $PResTrans: \Psi \otimes \Psi_Q$
 $\triangleright P \mapsto RBrOut M ((\nu*xvec)N \prec' P')$
by (*simp add: residualInject*)

note $QTrans$
moreover from $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$ **have** $x \# \Psi \otimes \Psi_P$ **by** *force*
moreover note $\langle x \# M \rangle \langle x \# N \rangle$
moreover from $QTrans \langle x \# N \rangle$ **have** $x \# xQ'$ **by** (*force dest: brinput-FreshDerivative simp add: abs-fresh*)
ultimately show *?case*
proof (*induct rule: resBrInputCases*)
case (*cRes Q'*)
have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q'$ **by** *fact*
with $\langle A_Q \#* Q \rangle \langle A_Q \#* N \rangle$ **have** $A_Q \#* Q'$
by (*elim brinputFreshChainDerivative*)

with $\langle extractFrame((\nu x)Q) = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $Fr xQ: (\nu x)(extractFrame Q) = \langle A_Q, \Psi_Q \rangle$ **by** *simp*
then obtain $y A_Q'$ **where** $A: A_Q = y\#A_Q'$ **by** (*cases A_Q auto*)
with $\langle A_Q \#* \Psi \rangle \langle A_Q \#* P \rangle \langle A_Q \#* P' \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* xvec \rangle$
 $\langle A_Q \#* M \rangle \langle A_Q \#* Q' \rangle \langle A_Q \#* N \rangle$
have $A_Q' \#* \Psi$ **and** $A_Q' \#* P$ **and** $A_Q' \#* \Psi_P$ **and** $A_Q' \#* Q$ **and** $A_Q \#*$
 $xvec$ **and** $A_Q \#* Q'$
and $y \# \Psi$ **and** $y \# P$ **and** $y \# P'$ **and** $y \# \Psi_P$ **and** $y \# Q$ **and** $y \# xvec$
and $y \# M$ **and** $y \# Q'$ **and** $y \# N$
and $A_Q' \#* M$
by (*simp*)+
from $A \langle A_P \#* A_Q \rangle$ **have** $A_P \#* A_Q'$ **and** $y \# A_P$ **by** (*simp add: fresh-star-list-cons*)+
from $A \langle A_Q \#* x \rangle$ **have** $x \neq y$ **and** $x \# A_Q'$ **by** (*simp add: fresh-list-cons*)+

with $A \langle distinct A_Q \rangle$ **have** $y \# A_Q'$
by (*induct A_Q' (auto simp add: fresh-list-nil fresh-list-cons)*)

note $PTrans FrP$
moreover from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto iM(N) \prec Q' \rangle$ **have** $([(x, y)] \cdot (\Psi \otimes \Psi_P)) \triangleright ([(x, y)] \cdot Q) \mapsto ([(x, y)] \cdot (iM(N) \prec Q'))$
by (*rule semantics.eqt*)
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# \Psi_P \rangle \langle y \# \Psi_P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# N \rangle \langle y \# N \rangle$
have $\Psi \otimes \Psi_P \triangleright ([(x, y)] \cdot Q) \mapsto iM(N) \prec ([(x, y)] \cdot Q')$
by (*simp add: eqvts*)
moreover from $A \langle A_Q \#* x \rangle Fr xQ$ **have** $Fr Q: extractFrame([(x, y)] \cdot Q) = \langle A_Q', \Psi_Q \rangle$ **and** $y \# extractFrame Q$
by (*clarsimp simp add: alpha' eqvts frame.inject fresh-list-cons name-swap*)+
moreover from $\langle A_P \#* Q \rangle$ **have** $([(x, y)] \cdot A_P) \#* ([(x, y)] \cdot Q)$ **by** (*simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]*)
with $\langle A_P \#* x \rangle \langle y \# A_P \rangle$ **have** $A_P \#* ([(x, y)] \cdot Q)$ **by** *simp*
moreover from $\langle A_Q' \#* Q \rangle$ **have** $([(x, y)] \cdot A_Q') \#* ([(x, y)] \cdot Q)$ **by** (*simp*)

add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]
with $\langle x \# A_Q' \rangle \langle y \# A_Q' \rangle$ **have** $A_Q' \#* ((x, y) \cdot Q)$ **by** *simp*
moreover from $\langle xvec \#* Q \rangle$ **have** $((x, y) \cdot xvec) \#* ((x, y) \cdot Q)$ **by** (*simp*
add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with $\langle x \# xvec \rangle \langle y \# xvec \rangle$ **have** $xvec \#* ((x, y) \cdot Q)$ **by** *simp*
ultimately have $\Psi \triangleright (P \parallel ((x, y) \cdot Q)) \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (P' \parallel ((x, y) \cdot Q'))$
using $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle \langle A_P \#* A_Q' \rangle \langle A_Q' \#* \Psi \rangle \langle A_Q' \#* P \rangle \langle A_P \#* M \rangle \langle A_Q' \#* M \rangle$
by (*intro BrComm2*) (*assumption* | *simp*) +
with $\langle y \# \Psi \rangle \langle y \# M \rangle \langle y \# xvec \rangle \langle y \# N \rangle$ **have** $\Psi \triangleright (\nu y) (P \parallel ((x, y) \cdot Q)) \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (\nu y) (P' \parallel ((x, y) \cdot Q'))$
by (*intro Scope*) *auto*
moreover from $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle$ **have** $(\nu y) (P \parallel ((x, y) \cdot Q)) = (\nu x) ((x, y) \cdot (P \parallel ((x, y) \cdot Q)))$
by (*subst alphaRes[of x]*) (*auto simp add: calc-atm fresh-left name-swap*)
with $\langle x \# P \rangle \langle y \# P \rangle$ **have** $(\nu y) (P \parallel ((x, y) \cdot Q)) = (\nu x) (P \parallel Q)$
by (*simp add: eqvts*)
moreover from $\langle x \# P' \rangle \langle y \# P' \rangle \langle y \# Q' \rangle \langle x \# xvec \rangle \langle y \# xvec \rangle$ **have** $(\nu y) (P' \parallel ((x, y) \cdot Q')) = (\nu x) (P' \parallel Q')$
by (*subst alphaRes[of y]*) (*auto simp add: resChainFresh calc-atm eqvts fresh-left name-swap*)
ultimately have $\Psi \triangleright (\nu x) (P \parallel Q) \mapsto_{iM} (\nu * xvec) \langle N \rangle \prec (\nu x) (P' \parallel Q')$
by *simp*
moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle$ **have** $(\Psi, (\nu * [x]) (P' \parallel Q'), (P' \parallel ((\nu * [x]) Q'))) \in Rel$
by (*intro C4*) *simp* +
moreover then have $(\Psi, (\nu x) (P' \parallel Q'), (P' \parallel ((\nu x) Q'))) \in Rel$
by *simp*
ultimately show *?case by blast*
qed
qed
qed
qed

lemma *scopeExtRight*:

fixes $x :: name$
and $P :: ('a, 'b, 'c) psi$
and $\Psi :: 'b$
and $Q :: ('a, 'b, 'c) psi$
and $Rel :: ('b \times ('a, 'b, 'c) psi \times ('a, 'b, 'c) psi) set$

assumes $x \# P$

and $x \# \Psi$

and *eqvt Rel*

and $C1: \bigwedge \Psi' R. (\Psi, R, R) \in Rel$

and $C2: \bigwedge y \Psi' R S zvec. \llbracket y \# \Psi'; y \# R; zvec \#* \Psi \rrbracket \implies (\Psi', (\nu * zvec) (R \parallel (\nu y) S), (\nu y) ((\nu * zvec) (R \parallel S))) \in Rel$

— Addition for Broadcast

and $C3: \bigwedge \Psi' R S \text{ zvec. } \llbracket \text{zvec} \#* R; \text{zvec} \#* \Psi' \rrbracket \implies (\Psi', (R \parallel (\nu * \text{zvec}) S)),$
 $((\nu * \text{zvec})(R \parallel S)) \in \text{Rel}$

shows $\Psi \triangleright P \parallel (\nu x) Q \rightsquigarrow[\text{Rel}] (\nu x)(P \parallel Q)$

proof –

note $\langle \text{eqvt Rel} \rangle \langle x \# \Psi \rangle$
moreover from $\langle x \# P \rangle$ **have** $x \# P \parallel (\nu x) Q$ **by** $(\text{simp add: abs-fresh})$
moreover from $\langle x \# P \rangle$ **have** $x \# (\nu x)(P \parallel Q)$ **by** $(\text{simp add: abs-fresh})$
ultimately show $?thesis$
proof $(\text{induct rule: simIFresh}[of \text{---} ()])$
case $(cSim \alpha xPQ)$
from $\langle bn \alpha \#* (P \parallel (\nu x) Q) \rangle \langle x \# \alpha \rangle$ **have** $bn \alpha \#* P$ **and** $bn \alpha \#* Q$ **by**
 simp+
note $\langle \Psi \triangleright (\nu x)(P \parallel Q) \mapsto \alpha \prec xPQ \rangle \langle x \# \Psi \rangle \langle x \# \alpha \rangle \langle x \# xPQ \rangle \langle bn \alpha \#* \Psi \rangle$
moreover from $\langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle$ **have** $bn \alpha \#* (P \parallel Q)$ **by** simp
ultimately show $?case$ **using** $\langle bn \alpha \#* \text{subject } \alpha \rangle \langle x \# \alpha \rangle$
proof $(\text{induct rule: resCases}[\text{where } C=()])$
case $(cOpen M \text{ xvec1 } \text{ xvec2 } y N PQ)$
from $\langle x \# M(\nu*(\text{xvec1}@y\#\text{xvec2}))\langle N \rangle \rangle$ **have** $x \# \text{xvec1}$ **and** $x \neq y$ **and** $x \#$
 xvec2 **and** $x \# M$ **by** simp+
from $\langle \text{xvec1} \#* (P \parallel Q) \rangle \langle \text{xvec2} \#* (P \parallel Q) \rangle \langle y \# (P \parallel Q) \rangle$
have $(\text{xvec1}@xvec2) \#* P$ **and** $(\text{xvec1}@xvec2) \#* Q$ **and** $y \# P$ **and** $y \# Q$
by simp+
from $\langle \Psi \triangleright P \parallel Q \mapsto M(\nu*(\text{xvec1}@xvec2))\langle [(x, y)] \cdot N \rangle \prec [(x, y)] \cdot PQ \rangle$
have $\langle [(x, y)] \cdot \Psi \triangleright [(x, y)] \cdot (P \parallel Q) \mapsto [(x, y)] \cdot (M(\nu*(\text{xvec1}@xvec2))\langle [(x,$
 $y) \cdot N \rangle \prec [(x, y)] \cdot PQ) \rangle$
by $(\text{rule semantics.eqvt})$
with $\langle x \# \Psi \rangle \langle y \# \Psi \rangle \langle x \# P \rangle \langle y \# P \rangle \langle x \# M \rangle \langle y \# M \rangle \langle x \# \text{xvec1} \rangle \langle x \# \text{xvec2} \rangle$
 $\langle y \# \text{xvec1} \rangle \langle y \# \text{xvec2} \rangle$
have $\Psi \triangleright P \parallel ((x, y) \cdot Q) \mapsto M(\nu*(\text{xvec1}@xvec2))\langle N \rangle \prec PQ$
by (simp add: eqvts)
moreover from $\langle \text{xvec1} \#* \Psi \rangle \langle \text{xvec2} \#* \Psi \rangle$ **have** $(\text{xvec1}@xvec2) \#* \Psi$ **by**
 simp
moreover note $\langle (\text{xvec1}@xvec2) \#* P \rangle$
moreover from $\langle (\text{xvec1}@xvec2) \#* Q \rangle$ **have** $\langle [(x, y)] \cdot (\text{xvec1}@xvec2) \rangle \#*$
 $\langle [(x, y)] \cdot Q \rangle$
by $(\text{simp add: pt-fresh-star-bij}[OF \text{pt-name-inst}, OF \text{at-name-inst}])$
with $\langle x \# \text{xvec1} \rangle \langle x \# \text{xvec2} \rangle \langle y \# \text{xvec1} \rangle \langle y \# \text{xvec2} \rangle$ **have** $(\text{xvec1}@xvec2) \#*$
 $\langle [(x, y)] \cdot Q \rangle$
by $(\text{auto simp add: eqvts})$
moreover from $\langle \text{xvec1} \#* M \rangle \langle \text{xvec2} \#* M \rangle$ **have** $(\text{xvec1}@xvec2) \#* M$ **by**
 simp
ultimately show $?case$
proof $(\text{induct rule: parOutputCases}[\text{where } C=y])$
case $(cPar1 P' A_Q \Psi_Q)$
from $\langle y \# \text{xvec1} \rangle \langle y \# \text{xvec2} \rangle$ **have** $y \# (\text{xvec1}@xvec2)$ **by** $(\text{auto simp add:}$
 $\text{fresh-list-append})$
with $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu*(\text{xvec1}@xvec2))\langle N \rangle \prec P' \rangle \langle (\text{xvec1}@xvec2) \#*$
 $M \rangle \langle y \# P \rangle$

```

    ‹distinct xvec1› ‹distinct xvec2› ‹xvec1 #* xvec2›
  have y # N by(force intro: outputFreshDerivative)
  with ‹y ∈ supp N› have False by(simp add: fresh-def)
  then show ?case by simp
next
case(cPar2 Q' AP ΨP)
have FrP: extractFrame P = ‹AP, ΨP› by fact
with ‹y # P› ‹AP #* y› have y # ΨP
  apply(drule-tac extractFrameFresh)
  by(simp add: frameResChainFresh) (simp add: fresh-def name-list-supp)
from ‹Ψ ⊗ ΨP ▷ ([[x, y]] · Q) ⟶ M(ν*(xvec1@xvec2))⟨N⟩ < Q'› ‹y ∈
supp N› ‹y # Ψ› ‹y # ΨP› ‹y # M› ‹y # xvec1› ‹y # xvec2›
  have Ψ ⊗ ΨP ▷ (νy)([[x, y]] · Q) ⟶ M(ν*(xvec1@y#xvec2))⟨N⟩ < Q'
by(force intro: Open)
  with ‹y # Q› have Ψ ⊗ ΨP ▷ (νx) Q ⟶ M(ν*(xvec1@y#xvec2))⟨N⟩ <
Q'
  by(simp add: alphaRes)
  moreover from ‹AP #* ([[x, y]] · Q)› have AP #* (νy)([[x, y]] · Q)
by(auto simp add: fresh-star-def abs-fresh)
  with ‹y # Q› have AP #* (νx) Q by(simp add: alphaRes)
  ultimately have Ψ ▷ P || ((νx) Q) ⟶ M(ν*(xvec1@y#xvec2))⟨N⟩ < (P
|| Q')
  using FrP ‹(xvec1@xvec2) #* P› ‹AP #* Ψ› ‹AP #* M› ‹y # P› ‹AP #*
(xvec1@xvec2)› ‹AP #* y› ‹AP #* N›
  by(intro Par2) auto
  moreover have (Ψ, P || Q', P || Q') ∈ Rel by(rule C1)
  ultimately show ?case by blast
qed
next
case(cBrOpen M xvec1 xvec2 y N PQ)
from ‹x # iM(ν*(xvec1@y#xvec2))⟨N⟩› have x # xvec1 and x ≠ y and x #
xvec2 and x # M by simp+
from ‹xvec1 #* (P || Q)› ‹xvec2 #* (P || Q)› ‹y # (P || Q)›
  have (xvec1@xvec2) #* P and (xvec1@xvec2) #* Q and y # P and y # Q
  by simp+
from ‹Ψ ▷ P || Q ⟶ iM(ν*(xvec1@xvec2))⟨([[x, y]] · N)⟩ < ([[x, y]] ·
PQ)›
  have ([[x, y]] · Ψ) ▷ ([[x, y]] · (P || Q)) ⟶ ([[x, y]] · (iM(ν*(xvec1@xvec2))⟨([[x,
y]] · N)⟩ < ([[x, y]] · PQ)))
  by(rule semantics.eqvt)
  with ‹x # Ψ› ‹y # Ψ› ‹x # P› ‹y # P› ‹x # M› ‹y # M› ‹x # xvec1› ‹x # xvec2›
  ‹y # xvec1› ‹y # xvec2›
  have Ψ ▷ P || ([[x, y]] · Q) ⟶ iM(ν*(xvec1@xvec2))⟨N⟩ < PQ
  by(simp add: eqvts)
  moreover from ‹xvec1 #* Ψ› ‹xvec2 #* Ψ› have (xvec1@xvec2) #* Ψ by
simp
  moreover note ‹(xvec1@xvec2) #* P›
  moreover from ‹(xvec1@xvec2) #* Q› have ([[x, y]] · (xvec1@xvec2)) #*
([[x, y]] · Q)

```

by(*simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*])
with $\langle x \# xvec1 \rangle \langle x \# xvec2 \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ **have** $(xvec1 @ xvec2) \#*$
 $([(x, y)] \cdot Q)$
by(*auto simp add: eqvts*)
moreover from $\langle xvec1 \#* M \rangle \langle xvec2 \#* M \rangle$ **have** $(xvec1 @ xvec2) \#* M$ **by**
simp
ultimately show ?*case*
proof(*induct rule: parBrOutputCases*[**where** $C=y$])
case(*cPar1 P' A_Q Ψ_Q*)
from $\langle y \# xvec1 \rangle \langle y \# xvec2 \rangle$ **have** $y \# (xvec1 @ xvec2)$ **by**(*auto simp add:*
fresh-list-append)
with $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu*(xvec1 @ xvec2)) \langle N \rangle \prec P' \rangle \langle (xvec1 @ xvec2)$
 $\#* M \rangle \langle y \# P \rangle$
 $\langle distinct\ xvec1 \rangle \langle distinct\ xvec2 \rangle \langle xvec1 \#* xvec2 \rangle$
have $y \# N$ **by**(*force intro: broutputFreshDerivative*)
with $\langle y \in supp\ N \rangle$ **have** *False* **by**(*simp add: fresh-def*)
then show ?*case by simp*
next
case(*cPar2 Q' A_P Ψ_P*)
have *FrP: extractFrame P = ⟨A_P, Ψ_P⟩* **by fact**
with $\langle y \# P \rangle \langle A_P \#* y \rangle$ **have** $y \# \Psi_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from $\langle \Psi \otimes \Psi_P \triangleright [(x, y)] \cdot Q \mapsto_i M(\nu*(xvec1 @ xvec2)) \langle N \rangle \prec Q' \rangle \langle y \in$
*supp N \rangle \langle y \# \Psi \rangle \langle y \# \Psi_P \rangle \langle y \# M \rangle \langle y \# xvec1 \rangle \langle y \# xvec2 \rangle
have $\Psi \otimes \Psi_P \triangleright (\nu y)([(x, y)] \cdot Q) \mapsto_i M(\nu*(xvec1 @ y \# xvec2)) \langle N \rangle \prec Q'$
by(*force intro: BrOpen*)
with $\langle y \# Q \rangle$ **have** $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto_i M(\nu*(xvec1 @ y \# xvec2)) \langle N \rangle \prec$
 Q'
by(*simp add: alphaRes*)
moreover from $\langle A_P \#* [(x, y)] \cdot Q \rangle$ **have** $A_P \#* (\nu y)([(x, y)] \cdot Q)$
by(*auto simp add: fresh-star-def abs-fresh*)
with $\langle y \# Q \rangle$ **have** $A_P \#* (\nu x)Q$ **by**(*simp add: alphaRes*)
ultimately have $\Psi \triangleright P \parallel ((\nu x)Q) \mapsto_i M(\nu*(xvec1 @ y \# xvec2)) \langle N \rangle \prec (P$
 $\parallel Q')$
using *FrP* $\langle (xvec1 @ xvec2) \#* P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* M \rangle \langle y \# P \rangle \langle A_P \#*$
 $(xvec1 @ xvec2) \rangle \langle A_P \#* y \rangle \langle A_P \#* N \rangle$
by(*intro Par2*) *auto*
moreover have $(\Psi, P \parallel Q', P \parallel Q') \in Rel$ **by**(*rule C1*)
ultimately show ?*case by blast*
next
case(*cBrComm1 Ψ_Q P' A_P Ψ_P Q' A_Q*)
have *FrP: extractFrame P = ⟨A_P, Ψ_P⟩* **by fact**
with $\langle y \# P \rangle \langle A_P \#* y \rangle$ **have** $y \# \Psi_P$
apply(*drule-tac extractFrameFresh*)
by(*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)

from $\langle extractFrame [(x, y)] \cdot Q = \langle A_Q, \Psi_Q \rangle \rangle$
have $extractFrame ((\nu y)([(x, y)] \cdot Q)) = \langle (y \# A_Q), \Psi_Q \rangle$*

```

    by simp
  with ⟨y # Q⟩
  have FrQ: extractFrame ((νx) Q) = ⟨(y#AQ), ΨQ⟩
    by(simp add: alphaRes)
  from ⟨Ψ ⊗ ΨP ▷ [(x, y)] · Q ⟶iM(ν*(xvec1 @ xvec2))⟨N⟩ < Q'⟩ ⟨y ∈
  supp N⟩ ⟨y # Ψ⟩ ⟨y # ΨP⟩ ⟨y # M⟩ ⟨y # xvec1⟩ ⟨y # xvec2⟩
  have Ψ ⊗ ΨP ▷ (νy)((x, y) · Q) ⟶iM(ν*(xvec1@y#xvec2))⟨N⟩ < Q'
    by(force intro: BrOpen)
  with ⟨y # Q⟩ have QTrans: Ψ ⊗ ΨP ▷ (νx) Q ⟶iM(ν*(xvec1@y#xvec2))⟨N⟩
  < Q'
    by(simp add: alphaRes)
  from ⟨AP #* ((x, y) · Q)⟩ have AP #* (νy)((x, y) · Q) by(auto simp
  add: fresh-star-def abs-fresh)
  with ⟨y # Q⟩ have AP #* (νx) Q by(simp add: alphaRes)
  from ⟨AQ #* ((x, y) · Q)⟩ have AQ #* (νy)((x, y) · Q) by(auto simp
  add: fresh-star-def abs-fresh)
  with ⟨y # Q⟩ have AQ #* (νx) Q by(simp add: alphaRes)
  moreover from ⟨y # Q⟩ have y # (νx) Q
    by (metis resChain.base resChain.step resChainFresh)
  ultimately have (y # AQ) #* ((νx) Q) by simp
  from ⟨Ψ ⊗ ΨQ ▷ P ⟶iM(N) < P'⟩ FrP QTrans FrQ
  ⟨AP #* Ψ⟩ ⟨AP #* P⟩ ⟨AP #* (νx) Q⟩ ⟨AP #* M⟩ ⟨AP #* y⟩ ⟨AP #* AQ⟩
  ⟨y # Ψ⟩ ⟨AQ #* Ψ⟩ ⟨y # P⟩ ⟨AQ #* P⟩ ⟨(y#AQ) #* (νx) Q⟩
  ⟨y # M⟩ ⟨AQ #* M⟩ ⟨y # P⟩ ⟨(xvec1@xvec2) #* P⟩
  have Ψ ▷ P || ((νx) Q) ⟶iM(ν*(xvec1@y#xvec2))⟨N⟩ < (P' || Q')
    by(intro BrComm1) (assumption | simp)+
  moreover have (Ψ, P' || Q', P' || Q') ∈ Rel by(rule C1)
  ultimately show ?case by blast
next
case(cBrComm2 ΨQ P' AP ΨP Q' AQ)
  from ⟨y # xvec1⟩ ⟨y # xvec2⟩ have y # (xvec1@xvec2) by(auto simp add:
  fresh-list-append)
  with ⟨Ψ ⊗ ΨQ ▷ P ⟶iM(ν*(xvec1@xvec2))⟨N⟩ < P'⟩ ⟨(xvec1@xvec2)
  #* M⟩ ⟨y # P⟩
  ⟨distinct xvec1⟩ ⟨distinct xvec2⟩ ⟨xvec1 #* xvec2⟩
  have y # N by(force intro: broutputFreshDerivative)
  with ⟨y ∈ supp N⟩ have False by(simp add: fresh-def)
  then show ?case by simp
qed
next
case(cRes PQ)
  from ⟨Ψ ▷ P || Q ⟶α < PQ⟩ ⟨bn α #* Ψ⟩ ⟨bn α #* P⟩ ⟨bn α #* Q⟩ ⟨bn α
  #* subject α⟩
  show ?case
  proof(induct rule: parCases[where C=x])
    case(cPar1 P' AQ ΨQ)
      note ⟨Ψ ⊗ ΨQ ▷ P ⟶α < P'⟩
      moreover with ⟨x # P⟩ ⟨x # α⟩ ⟨bn α #* subject α⟩ ⟨distinct(bn α)⟩
      have x # P' by(force dest: freeFreshDerivative)

```

with $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $\text{extractFrame}(\langle \nu x \rangle Q) = \langle \langle x \# A_Q \rangle, \Psi_Q \rangle$
by *simp*
moreover from $\langle \text{bn } \alpha \#* Q \rangle$ **have** $\text{bn } \alpha \#* (\langle \nu x \rangle Q)$ **by** (*simp add: fresh-star-def abs-fresh*)
moreover from $\langle x \# \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $\langle x \# A_Q \rangle \#* \Psi$ **by** *simp*
moreover from $\langle x \# P \rangle \langle A_Q \#* P \rangle$ **have** $\langle x \# A_Q \rangle \#* P$ **by** *simp*
moreover from $\langle x \# \alpha \rangle \langle A_Q \#* \alpha \rangle$ **have** $\langle x \# A_Q \rangle \#* \alpha$ **by** *simp*
ultimately have $\Psi \triangleright P \parallel \langle \nu x \rangle Q \mapsto \alpha \prec (P' \parallel \langle \nu x \rangle Q)$
by (*rule Par1*)
moreover from $\langle x \# P' \rangle \langle x \# \Psi \rangle$ **have** $(\Psi, \langle \nu * \rangle(P' \parallel \langle \nu x \rangle Q)), \langle \nu x \rangle(\langle \nu * \rangle(P' \parallel Q)) \in \text{Rel}$
by (*intro C2*) *auto*
ultimately show *?case*
by *force*
next
case (*cPar2 Q' A_P \Psi_P*)
have *FrP: extractFrame P = \langle A_P, \Psi_P \rangle* **by** *fact*
with $\langle x \# P \rangle \langle A_P \#* x \rangle$ **have** $x \# \Psi_P$
apply (*drule-tac extractFrameFresh*)
by (*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle \langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# \alpha \rangle$
have $\Psi \otimes \Psi_P \triangleright \langle \nu x \rangle Q \mapsto \alpha \prec \langle \nu x \rangle Q'$
by (*intro Scope*) *auto*
moreover note *FrP \langle \text{bn } \alpha \#* P \rangle \langle A_P \#* \Psi \rangle*
moreover from $\langle A_P \#* Q \rangle$ **have** $A_P \#* \langle \nu x \rangle Q$ **by** (*simp add: fresh-star-def abs-fresh*)
ultimately have $\Psi \triangleright P \parallel \langle \nu x \rangle Q \mapsto \alpha \prec (P \parallel \langle \nu x \rangle Q')$ **using** $\langle A_P \#* \alpha \rangle$
by (*rule Par2*)
moreover from $\langle x \# P \rangle \langle x \# \Psi \rangle$ **have** $(\Psi, \langle \nu * \rangle(P \parallel \langle \nu x \rangle Q')), \langle \nu x \rangle(\langle \nu * \rangle(P \parallel Q')) \in \text{Rel}$
by (*intro C2*) *auto*
ultimately show *?case*
by *force*
next
case (*cComm1 \Psi_Q M N P' A_P \Psi_P K xvec Q' A_Q*)
have *PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto M \langle N \rangle \prec P'* **and** *FrP: extractFrame P = \langle A_P, \Psi_P \rangle* **by** *fact+*
have *QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto K \langle \nu * xvec \rangle \langle N \rangle \prec Q'* **and** *FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle* **by** *fact+*
from *FrP \langle x \# P \rangle* **have** $x \# \langle A_P, \Psi_P \rangle$
by (*auto dest: extractFrameFresh*)
with $\langle A_P \#* x \rangle$ **have** $x \# \Psi_P$ **by** (*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from *PTrans FrP \langle distinct A_P \rangle \langle x \# P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* x \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle*
obtain M' **where** *MeqM': (\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'* **and** $x \# M'$ **and** $A_Q \#* M'$
by (*elim inputObtainPrefix[where B=x\#A_Q]*) (*assumption | force simp*)

```

add: fresh-star-list-cons)+
  from MeqM' have MeqM':  $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$ 
    by(metis statEqEnt Associativity Composition Commutativity)
  with  $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$  have  $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$ 
    by(blast intro: chanEqTrans chanEqSym)
  then have  $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$ 
    by(metis statEqEnt Associativity Composition Commutativity)
  with QTrans FrQ  $\langle \text{distinct } A_Q \rangle$  have QTrans:  $\Psi \otimes \Psi_P \triangleright Q \mapsto M'(\nu^*xvec)\langle N \rangle$ 
  < Q' using  $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$ 
    by(elim outputRenameSubject) (assumption | force)+
  show ?case
  proof(cases  $x \in \text{supp } N$ )
    note PTrans FrP
    moreover assume  $x \in \text{supp } N$ 
    then have  $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto M'(\nu^*([]@(\nu x\#xvec)))\langle N \rangle$  < Q' using
  QTrans  $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M' \rangle \langle xvec \#* x \rangle$ 
    by(intro Open) (assumption | force simp add: fresh-list-nil)+
    then have QTrans:  $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto M'(\nu^*(x\#xvec))\langle N \rangle$  < Q' by
  simp
    moreover from  $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$  have  $\text{extractFrame}((\nu x)Q)$ 
  =  $\langle (x\#A_Q), \Psi_Q \rangle$ 
    by simp
    moreover note MeqM'  $\langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle$ 
    moreover from  $\langle A_P \#* Q \rangle$  have  $A_P \#* ((\nu x)Q)$  by(simp add:
  fresh-star-def abs-fresh)
    moreover from  $\langle A_P \#* A_Q \rangle \langle A_P \#* x \rangle$  have  $A_P \#* (x\#A_Q)$ 
    by(simp add: fresh-star-def fresh-list-cons) (auto simp add: fresh-def
  name-list-supp)
    moreover from  $\langle x \# \Psi \rangle \langle A_Q \#* \Psi \rangle$  have  $(x\#A_Q) \#* \Psi$  by simp
    moreover from  $\langle x \# P \rangle \langle A_Q \#* P \rangle$  have  $(x\#A_Q) \#* P$  by simp
    moreover from  $\langle x \# M' \rangle \langle A_Q \#* M' \rangle$  have  $(x\#A_Q) \#* M'$  by simp
    moreover from  $\langle A_Q \#* Q \rangle$  have  $(x\#A_Q) \#* ((\nu x)Q)$  by(simp add:
  fresh-star-def abs-fresh)
    moreover from  $\langle x \# P \rangle \langle xvec \#* P \rangle$  have  $(x\#xvec) \#* P$  by(simp)
    ultimately have  $\Psi \triangleright P \parallel (\nu x)Q \mapsto \tau$  <  $(\nu^*(x\#xvec))\langle P' \parallel Q' \rangle$  using
   $\langle A_P \#* M \rangle$ 
    by(intro Comm1) (assumption | simp)+
    moreover have  $(\Psi, (\nu x)((\nu^*xvec)\langle P' \parallel Q' \rangle), (\nu x)((\nu^*xvec)\langle P' \parallel Q' \rangle))$ 
   $\in \text{Rel}$  by(rule C1)
    ultimately show ?case by force
  next
  note PTrans FrP
  moreover assume  $x \notin \text{supp } N$ 
  then have  $x \# N$  by(simp add: fresh-def)
  with QTrans  $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M' \rangle \langle xvec \#* x \rangle$ 
  have QTrans:  $\Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto M'(\nu^*xvec)\langle N \rangle$  <  $(\nu x)Q'$ 
    by(intro Scope) (assumption | force)+
    moreover from PTrans  $\langle x \# P \rangle \langle x \# N \rangle$  have  $x \# P'$  by(rule input-
  FreshDerivative)

```


with $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $\text{extractFrame}(\nu x)Q = \langle (x\#A_Q), \Psi_Q \rangle$
by *simp*
moreover note $\text{MeqM}' \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle$
moreover from $\langle A_P \#* Q \rangle$ **have** $A_P \#* (\nu x)Q$ **by**(*simp add: fresh-star-def abs-fresh*)
moreover from $\langle A_P \#* A_Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* (x\#A_Q)$
by(*simp add: fresh-star-def fresh-list-cons*) (*auto simp add: fresh-def name-list-supp*)
moreover from $\langle x \# \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $(x\#A_Q) \#* \Psi$ **by** *simp*
moreover from $\langle x \# P \rangle \langle A_Q \#* P \rangle$ **have** $(x\#A_Q) \#* P$ **by** *simp*
moreover from $\langle x \# M' \rangle \langle A_Q \#* M' \rangle$ **have** $(x\#A_Q) \#* M'$ **by** *simp*
moreover from $\langle A_Q \#* Q \rangle$ **have** $(x\#A_Q) \#* (\nu x)Q$ **by**(*simp add: fresh-star-def abs-fresh*)
moreover from $\langle x \# P \rangle \langle \text{vec} \#* P \rangle$ **have** $(x\#\text{vec}) \#* P$ **by**(*simp*)
ultimately have $\Psi \triangleright P \parallel (\nu x)Q \mapsto \tau \prec (\nu * \text{vec})(P' \parallel (\nu x)Q')$ **using**
 $\langle A_P \#* M \rangle$
by(*intro Comm1*) (*assumption* | *simp*)+
moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle \langle \text{vec} \#* \Psi \rangle$ **have** $(\Psi, (\nu * \text{vec})(P' \parallel (\nu x)Q'), (\nu x)((\nu * \text{vec})(P' \parallel Q'))) \in \text{Rel}$ **by**(*rule C2*)
ultimately show *?case* **by** *blast*
qed
next
case(*cComm2* $\Psi_Q M \text{vec } N P' A_P \Psi_P K Q' A_Q$)
have $P\text{Trans}: \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu * \text{vec})(N) \prec P'$ **and** $\text{FrP}: \text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **by** *fact+*
have $Q\text{Trans}: \Psi \otimes \Psi_P \triangleright Q \mapsto K(N) \prec Q'$ **and** $\text{FrQ}: \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact+*
from $\text{FrP} \langle x \# P \rangle$ **have** $x \# \langle A_P, \Psi_P \rangle$
by(*auto dest: extractFrameFresh*)
with $\langle A_P \#* x \rangle$ **have** $x \# \Psi_P$ **by**(*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from $P\text{Trans}$
have $\Psi \otimes \Psi_Q \triangleright P \mapsto R\text{Out } M ((\nu * \text{vec})N \prec' P')$
by(*simp add: residualInject*)
with $\text{FrP} \langle \text{distinct } A_P \rangle \langle x \# P \rangle \langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_Q \rangle \langle A_P \#* x \rangle \langle A_P \#* P \rangle \langle A_P \#* M \rangle \langle A_Q \#* P \rangle \langle A_P \#* A_Q \rangle \langle \text{vec} \#* M \rangle \langle \text{distinct } \text{vec} \rangle$
obtain M' **where** $\text{MeqM}': (\Psi \otimes \Psi_Q) \otimes \Psi_P \vdash M \leftrightarrow M'$ **and** $x \# M'$ **and**
 $A_Q \#* M'$
by(*elim outputObtainPrefix*[**where** $B=x\#A_Q$]) (*assumption* | *force simp add: fresh-star-list-cons*)+
from MeqM' **have** $\text{MeqM}': \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with $\langle \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K \rangle$ **have** $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*blast intro: chanEqTrans chanEqSym*)
then have $(\Psi \otimes \Psi_P) \otimes \Psi_Q \vdash K \leftrightarrow M'$
by(*metis statEqEnt Associativity Commutativity Composition*)
with $Q\text{Trans} \text{FrQ} \langle \text{distinct } A_Q \rangle$ **have** $Q\text{Trans}: \Psi \otimes \Psi_P \triangleright Q \mapsto M'(N) \prec Q'$ **using** $\langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* K \rangle \langle A_Q \#* M' \rangle$

by(*auto intro: inputRenameSubject*)

from $PTrans \langle x \# P \rangle \langle xvec \#* x \rangle \langle distinct \ xvec \rangle \langle xvec \#* M \rangle$
have $x \# N$ **and** $x \# P'$ **by**(*force intro: outputFreshDerivative*)+
from $QTrans \langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M' \rangle \langle x \# N \rangle$ **have** $QTrans: \Psi \otimes \Psi_P$
 $\triangleright (\nu x)Q \mapsto M'(\downarrow N) \prec (\nu x)Q'$
by(*intro Scope*) (*assumption* | *force*)+

note $PTrans \ FrP \ QTrans$
moreover with $\langle extractFrame \ Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $extractFrame((\nu x)Q)$
 $= \langle (x\#A_Q), \Psi_Q \rangle$
by *simp*
moreover note $MeqM' \langle A_P \#* \Psi \rangle \langle A_P \#* P \rangle$
moreover from $\langle A_P \#* Q \rangle$ **have** $A_P \#* ((\nu x)Q)$ **by**(*simp add: fresh-star-def abs-fresh*)
moreover from $\langle A_P \#* A_Q \rangle \langle A_P \#* x \rangle$ **have** $A_P \#* (x\#A_Q)$
by(*simp add: fresh-star-def fresh-list-cons*) (*auto simp add: fresh-def name-list-supp*)
moreover from $\langle x \# \Psi \rangle \langle A_Q \#* \Psi \rangle$ **have** $(x\#A_Q) \#* \Psi$ **by** *simp*
moreover from $\langle x \# P \rangle \langle A_Q \#* P \rangle$ **have** $(x\#A_Q) \#* P$ **by** *simp*
moreover from $\langle x \# M' \rangle \langle A_Q \#* M' \rangle$ **have** $(x\#A_Q) \#* M'$ **by** *simp*
moreover from $\langle A_Q \#* Q \rangle$ **have** $(x\#A_Q) \#* ((\nu x)Q)$ **by**(*simp add: fresh-star-def abs-fresh*)
moreover from $\langle xvec \#* Q \rangle$ **have** $(x\#xvec) \#* ((\nu x)Q)$ **by**(*simp add: abs-fresh fresh-star-def*)
ultimately have $\Psi \triangleright P \parallel (\nu x)Q \mapsto \tau \prec (\nu *xvec)(P' \parallel (\nu x)Q')$ **using**
 $\langle A_P \#* M \rangle$
by(*intro Comm2*) (*assumption* | *simp*)+
moreover from $\langle x \# \Psi \rangle \langle x \# P' \rangle \langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu *xvec)(P' \parallel (\nu x)Q'), (\nu x)((\nu *xvec)(P' \parallel Q'))) \in Rel$ **by**(*rule C2*)
ultimately show *?case* **by** *blast*

next
case(*cBrMerge* $\Psi_Q \ M \ N \ P' \ A_P \ \Psi_P \ Q' \ A_Q$)
have $PTrans: \Psi \otimes \Psi_Q \triangleright P \mapsto \downarrow M(\downarrow N) \prec P'$ **and** $FrP: extractFrame \ P = \langle A_P, \Psi_P \rangle$ **by** *fact*+
have $QTrans: \Psi \otimes \Psi_P \triangleright Q \mapsto \downarrow M(\downarrow N) \prec Q'$ **and** $FrQ: extractFrame \ Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact*+
from $FrP \langle x \# P \rangle$ **have** $x \# \langle A_P, \Psi_P \rangle$
by(*auto dest: extractFrameFresh*)
with $\langle A_P \#* x \rangle$ **have** $x \# \Psi_P$
by(*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from $\langle x \# \alpha \rangle \langle \alpha = \downarrow M(\downarrow N) \rangle$ **have** $x \# M$ **and** $x \# N$ **by** *simp*+
from $PTrans \langle x \# P \rangle \langle x \# N \rangle$
have $x \# P'$
by(*rule brinputFreshDerivative*)
from $QTrans \langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M \rangle \langle x \# N \rangle$ **have** $QTrans: \Psi \otimes \Psi_P \triangleright (\nu x)Q \mapsto \downarrow M(\downarrow N) \prec (\nu x)Q'$
by(*intro Scope*) (*assumption* | *force*)+

```

note PTrans FrP QTrans
moreover with  $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$  have  $\text{extractFrame}(\langle \nu x \rangle Q)$ 
=  $\langle \langle x \# A_Q \rangle, \Psi_Q \rangle$ 
  by simp
  moreover note  $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle$ 
  moreover from  $\langle A_P \# Q \rangle$  have  $A_P \# \langle \langle \nu x \rangle Q \rangle$  by (simp add: fresh-star-def
abs-fresh)
  moreover from  $\langle A_P \# A_Q \rangle \langle A_P \# x \rangle$  have  $A_P \# \langle x \# A_Q \rangle$ 
    by (simp add: fresh-star-def fresh-list-cons) (auto simp add: fresh-def
name-list-supp)
  moreover from  $\langle x \# \Psi \rangle \langle A_Q \# \Psi \rangle$  have  $\langle x \# A_Q \rangle \# \Psi$  by simp
  moreover from  $\langle x \# P \rangle \langle A_Q \# P \rangle$  have  $\langle x \# A_Q \rangle \# P$  by simp
  moreover from  $\langle x \# M \rangle \langle A_Q \# M \rangle$  have  $\langle x \# A_Q \rangle \# M$  by simp
  moreover from  $\langle A_Q \# Q \rangle$  have  $\langle x \# A_Q \rangle \# \langle \langle \nu x \rangle Q \rangle$  by (simp add:
fresh-star-def abs-fresh)
  ultimately have Trans:  $\Psi \triangleright P \parallel \langle \nu x \rangle Q \mapsto_i M \langle N \rangle \prec (P' \parallel \langle \nu x \rangle Q')$ 
using  $\langle A_P \# M \rangle$ 
  by (intro BrMerge) (assumption | simp) +
  from  $\langle x \# \Psi \rangle \langle x \# P' \rangle$  have  $(\Psi, (P' \parallel \langle \nu * [x] \rangle Q'), \langle \nu * [x] \rangle (P' \parallel Q')) \in \text{Rel}$ 
  by (intro C3) simp +
  then have Relation:  $(\Psi, (P' \parallel \langle \nu x \rangle Q'), \langle \nu x \rangle (P' \parallel Q')) \in \text{Rel}$  by simp
  with Trans Relation show ?case by blast
next
  case (cBrComm1  $\Psi_Q M N P' A_P \Psi_P \text{xvec } Q' A_Q$ )
  from  $\langle x \# \alpha \rangle \langle_i M \langle \nu * \text{xvec} \rangle \langle N \rangle = \alpha \rangle$  have  $x \# M$  and  $x \# \text{xvec}$  and  $x \# N$ 
by force +
  have PTrans:  $\Psi \otimes \Psi_Q \triangleright P \mapsto_i M \langle N \rangle \prec P'$  and FrP:  $\text{extractFrame } P =$ 
 $\langle A_P, \Psi_P \rangle$  by fact +
  have QTrans:  $\Psi \otimes \Psi_P \triangleright Q \mapsto_i M \langle \nu * \text{xvec} \rangle \langle N \rangle \prec Q'$  and FrQ:  $\text{extractFrame}$ 
 $Q = \langle A_Q, \Psi_Q \rangle$  by fact +
  from FrP  $\langle x \# P \rangle$  have  $x \# \langle A_P, \Psi_P \rangle$ 
  by (auto dest: extractFrameFresh)
  with  $\langle A_P \# x \rangle$  have  $x \# \Psi_P$ 
  by (simp add: frameResChainFresh) (simp add: fresh-def name-list-supp)
  show ?case
  proof (cases  $x \in \text{supp } N$ )
  note PTrans FrP
  moreover assume  $x \in \text{supp } N$ 
  with  $\langle x \# N \rangle$  have False
  by (simp add: fresh-def)
  then show ?case
  by simp
next
  note PTrans FrP
  moreover assume  $x \notin \text{supp } N$ 
  from QTrans  $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M \rangle \langle x \# N \rangle \langle x \# \text{xvec} \rangle$ 
  have QTrans:  $\Psi \otimes \Psi_P \triangleright \langle \nu x \rangle Q \mapsto_i M \langle \nu * \text{xvec} \rangle \langle N \rangle \prec \langle \nu x \rangle Q'$ 
  by (intro Scope) (assumption | force) +
  moreover from PTrans  $\langle x \# P \rangle \langle x \# N \rangle$  have  $x \# P'$  by (rule brinput-

```

FreshDerivative)
with $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $\text{extractFrame}(\llbracket \nu x \rrbracket Q) = \langle (x \# A_Q), \Psi_Q \rangle$
by *simp*
moreover note $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle$
moreover from $\langle A_P \# Q \rangle$ **have** $A_P \# (\llbracket \nu x \rrbracket Q)$ **by** (*simp add: fresh-star-def abs-fresh*)
moreover from $\langle A_P \# A_Q \rangle \langle A_P \# x \rangle$ **have** $A_P \# (x \# A_Q)$
by (*simp add: fresh-star-def fresh-list-cons*) (*auto simp add: fresh-def name-list-supp*)
moreover from $\langle x \# \Psi \rangle \langle A_Q \# \Psi \rangle$ **have** $(x \# A_Q) \# \Psi$ **by** *simp*
moreover from $\langle x \# P \rangle \langle A_Q \# P \rangle$ **have** $(x \# A_Q) \# P$ **by** *simp*
moreover from $\langle x \# M \rangle \langle A_Q \# M \rangle$ **have** $(x \# A_Q) \# M$ **by** *simp*
moreover from $\langle A_Q \# Q \rangle$ **have** $(x \# A_Q) \# (\llbracket \nu x \rrbracket Q)$ **by** (*simp add: fresh-star-def abs-fresh*)
moreover from $\langle x \# P \rangle \langle \text{vec} \# P \rangle$ **have** $(x \# \text{vec}) \# P$ **by** (*simp*)
ultimately have $\text{Trans: } \Psi \triangleright P \parallel (\llbracket \nu x \rrbracket Q) \mapsto_i M(\nu * \text{vec})(N) \prec (P' \parallel (\llbracket \nu x \rrbracket Q'))$ **using** $\langle A_P \# M \rangle$
by (*intro BrComm1*) (*assumption | simp*) +
from $\langle x \# \Psi \rangle \langle x \# P' \rangle$ **have** $(\Psi, (P' \parallel (\llbracket \nu * [x] \rrbracket Q')), (\llbracket \nu * [x] \rrbracket (P' \parallel Q'))) \in \text{Rel}$
by (*intro C3*) *simp* +
then have $\text{Relation: } (\Psi, (P' \parallel (\llbracket \nu x \rrbracket Q')), (\llbracket \nu x \rrbracket (P' \parallel Q'))) \in \text{Rel}$ **by** *simp*
from Trans Relation **show** *?case* **by** *blast*
qed
next
case (*cBrComm2* $\Psi_Q M \text{vec } N P' A_P \Psi_P Q' A_Q$)
from $\langle x \# \alpha \rangle \langle_i M(\nu * \text{vec})(N) = \alpha \rangle$ **have** $x \# M$ **and** $x \# \text{vec}$ **and** $x \# N$
by *force* +
have $P\text{Trans: } \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu * \text{vec})(N) \prec P'$ **and** $\text{FrP: } \text{extractFrame } P = \langle A_P, \Psi_P \rangle$ **by** *fact* +
have $Q\text{Trans: } \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(N) \prec Q'$ **and** $\text{FrQ: } \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle$ **by** *fact* +
from $\text{FrP} \langle x \# P \rangle$ **have** $x \# \langle A_P, \Psi_P \rangle$
by (*auto dest: extractFrameFresh*)
with $\langle A_P \# x \rangle$ **have** $x \# \Psi_P$
by (*simp add: frameResChainFresh*) (*simp add: fresh-def name-list-supp*)
from $P\text{Trans} \langle x \# P \rangle \langle x \# \text{vec} \rangle \langle \text{distinct } \text{vec} \rangle \langle \text{vec} \# M \rangle$
have $x \# N$ **and** $x \# P'$ **by** (*force intro: broutputFreshDerivative*) +
from $Q\text{Trans} \langle x \# \Psi \rangle \langle x \# \Psi_P \rangle \langle x \# M \rangle \langle x \# N \rangle$ **have** $Q\text{Trans: } \Psi \otimes \Psi_P \triangleright (\llbracket \nu x \rrbracket Q) \mapsto_i M(N) \prec (\llbracket \nu x \rrbracket Q')$
by (*intro Scope*) (*assumption | force*) +

note $P\text{Trans FrP } Q\text{Trans}$
moreover with $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ **have** $\text{extractFrame}(\llbracket \nu x \rrbracket Q) = \langle (x \# A_Q), \Psi_Q \rangle$
by *simp*
moreover note $\langle A_P \# \Psi \rangle \langle A_P \# P \rangle$
moreover from $\langle A_P \# Q \rangle$ **have** $A_P \# (\llbracket \nu x \rrbracket Q)$ **by** (*simp add: fresh-star-def*)

```

abs-fresh)
  moreover from  $\langle A_P \#^* A_Q \rangle \langle A_P \#^* x \rangle$  have  $A_P \#^* (x \# A_Q)$ 
    by(simp add: fresh-star-def fresh-list-cons) (auto simp add: fresh-def
name-list-supp)
  moreover from  $\langle x \# \Psi \rangle \langle A_Q \#^* \Psi \rangle$  have  $(x \# A_Q) \#^* \Psi$  by simp
  moreover from  $\langle x \# P \rangle \langle A_Q \#^* P \rangle$  have  $(x \# A_Q) \#^* P$  by simp
  moreover from  $\langle x \# M \rangle \langle A_Q \#^* M \rangle$  have  $(x \# A_Q) \#^* M$  by simp
  moreover from  $\langle A_Q \#^* Q \rangle$  have  $(x \# A_Q) \#^* ((\nu x)Q)$  by(simp add:
fresh-star-def abs-fresh)
  moreover from  $\langle xvec \#^* Q \rangle$  have  $(x \# xvec) \#^* ((\nu x)Q)$  by(simp add:
abs-fresh fresh-star-def)
  ultimately have  $Trans: \Psi \triangleright P \parallel ((\nu x)Q) \mapsto_i M(\nu^* xvec) \langle N \rangle \prec (P' \parallel$ 
 $(\nu x)Q')$  using  $\langle A_P \#^* M \rangle$ 
    by(intro BrComm2) (assumption | simp)+
  from  $\langle x \# \Psi \rangle \langle x \# P' \rangle$  have  $(\Psi, (P' \parallel ((\nu^*[x])Q')), (\nu^*[x])(P' \parallel Q')) \in Rel$ 
    by(intro C3) simp+
  then have  $Relation: (\Psi, (P' \parallel ((\nu x)Q')), ((\nu x)(P' \parallel Q'))) \in Rel$  by simp
  from  $Trans$  Relation show ?case by blast
qed
next
case(cBrClose M xvec N PQ)
from  $\langle xvec \#^* (P \parallel Q) \rangle$  have  $xvec \#^* P$  and  $xvec \#^* Q$  by simp+
note  $\langle \Psi \triangleright P \parallel Q \mapsto_i M(\nu^* xvec) \langle N \rangle \prec PQ \rangle \langle xvec \#^* \Psi \rangle \langle xvec \#^* P \rangle \langle xvec$ 
 $\#^* Q \rangle \langle xvec \#^* M \rangle$ 
then show ?case
proof(induct rule: parBrOutputCases[where C=x])
case(cPar1 P' A_Q  $\Psi_Q$ )
from  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto_i M(\nu^* xvec) \langle N \rangle \prec P' \rangle \langle xvec \#^* M \rangle \langle x \# P \rangle$ 
have  $x \# M$ 
  by(rule brOutputFreshSubject)
with  $\langle x \in supp M \rangle$  have False
  by(simp add: fresh-def)
then show ?case
  by simp
next
case(cPar2 Q' A_P  $\Psi_P$ )
from  $\langle A_P \#^* x \rangle$  have  $x \# A_P$  by simp
with  $\langle x \# P \rangle \langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle$ 
have  $x \# \Psi_P$ 
  apply(drule-tac extractFrameFresh)
  by(simp add: frameResChainFresh) (simp add: fresh-def name-list-supp)
from  $\langle x \# \Psi \rangle \langle x \# \Psi_P \rangle$  have  $x \# (\Psi \otimes \Psi_P)$  by force
from  $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto_i M(\nu^* xvec) \langle N \rangle \prec Q' \rangle \langle x \in supp M \rangle \langle x \# (\Psi \otimes$ 
 $\Psi_P) \rangle$ 
have  $QTrans: \Psi \otimes \Psi_P \triangleright ((\nu x)Q) \mapsto \tau \prec ((\nu x)((\nu^* xvec)Q')$ 
  by(rule BrClose)
with  $\langle extractFrame P = \langle A_P, \Psi_P \rangle \rangle \langle A_P \#^* \Psi \rangle \langle x \# A_P \rangle \langle A_P \#^* Q \rangle$ 
have  $Trans: \Psi \triangleright (P \parallel ((\nu x)Q)) \mapsto \tau \prec (P \parallel ((\nu x)((\nu^* xvec)Q'))$ 
  by(intro Par2) (assumption | simp)+

```

```

from  $\langle x \# P \rangle \langle xvec \#* P \rangle$  have  $(x\#xvec) \#* P$  by simp
moreover from  $\langle x \# \Psi \rangle \langle xvec \#* \Psi \rangle$  have  $(x\#xvec) \#* \Psi$  by simp
ultimately have  $(\Psi, (P \parallel (\nu*(x\#xvec))Q'), (\nu*(x\#xvec))P \parallel Q') \in$ 
Rel
  by(rule C3)
then have Relation:  $(\Psi, (P \parallel (\nu x)(\nu*xvec)Q'), (\nu x)(\nu*xvec)P \parallel Q')$ 
∈ Rel
  by simp
from Trans Relation
show ?case
  by blast
next
case(cBrComm1  $\Psi_Q P' A_P \Psi_P Q' A_Q$ )
from  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto iM(N) \prec P' \rangle \langle x \# P \rangle$ 
have  $x \# M$ 
  by(rule brInputFreshSubject)
with  $\langle x \in \text{supp } M \rangle$  have False
  by(simp add: fresh-def)
then show ?case
  by simp
next
case(cBrComm2  $\Psi_Q P' A_P \Psi_P Q' A_Q$ )
from  $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto iM(\nu*xvec)\langle N \rangle \prec P' \rangle \langle xvec \#* M \rangle \langle x \# P \rangle$ 
have  $x \# M$ 
  by(rule brOutputFreshSubject)
with  $\langle x \in \text{supp } M \rangle$  have False
  by(simp add: fresh-def)
then show ?case
  by simp
qed
qed
qed
qed

lemma simParComm:
  fixes  $\Psi :: 'b$ 
    and  $P :: ('a, 'b, 'c) \text{ psi}$ 
    and  $Q :: ('a, 'b, 'c) \text{ psi}$ 
    and  $Rel :: ('b \times ('a, 'b, 'c) \text{ psi} \times ('a, 'b, 'c) \text{ psi}) \text{ set}$ 

assumes eqvt Rel
  and  $C1: \bigwedge \Psi' R S. (\Psi', R \parallel S, S \parallel R) \in Rel$ 
  and  $C2: \bigwedge \Psi' R S xvec. \llbracket (\Psi', R, S) \in Rel; xvec \#* \Psi \rrbracket \implies (\Psi', (\nu*xvec)R,$ 
 $(\nu*xvec)S) \in Rel$ 

shows  $\Psi \triangleright P \parallel Q \rightsquigarrow[Rel] Q \parallel P$ 
  using  $\langle \text{eqvt } Rel \rangle$ 
proof(induct rule: simI[of - - - ()])
  case(cSim  $\alpha PQ$ )

```

from $\langle bn \ \alpha \ \#* \ (P \parallel Q) \rangle$ **have** $bn \ \alpha \ \#* \ Q$ **and** $bn \ \alpha \ \#* \ P$ **by** *simp+*
with $\langle \Psi \triangleright Q \parallel P \mapsto \alpha \prec PQ \rangle$ $\langle bn \ \alpha \ \#* \ \Psi \rangle$ **show** *?case using* $\langle bn \ \alpha \ \#* \ \text{subject} \ \alpha \rangle$
proof(*induct rule: parCases[where C=()]*)
 case(*cPar1 Q' A_P Ψ_P*)
 from $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q' \rangle$ $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$ $\langle bn \ \alpha \ \#* \ P \rangle$
 $\langle A_P \ \#* \ \Psi \rangle$ $\langle A_P \ \#* \ Q \rangle$ $\langle A_P \ \#* \ \alpha \rangle$
 have $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P \parallel Q')$ **by**(*rule Par2*)
 moreover have $(\Psi, P \parallel Q', Q' \parallel P) \in Rel$ **by**(*rule C1*)
 ultimately show *?case by blast*
 next
 case(*cPar2 P' A_Q Ψ_Q*)
 from $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P' \rangle$ $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$ $\langle bn \ \alpha \ \#* \ Q \rangle$
 $\langle A_Q \ \#* \ \Psi \rangle$ $\langle A_Q \ \#* \ P \rangle$ $\langle A_Q \ \#* \ \alpha \rangle$
 have $\Psi \triangleright P \parallel Q \mapsto \alpha \prec (P' \parallel Q)$ **by**(*rule Par1*)
 moreover have $(\Psi, P' \parallel Q, Q \parallel P') \in Rel$ **by**(*rule C1*)
 ultimately show *?case by blast*
 next
 case(*cComm1 Ψ_P M N Q' A_Q Ψ_Q K xvec P' A_P*)
 note $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto K(\nu * xvec) \langle N \rangle \prec P' \rangle$ $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M \langle N \rangle \prec Q' \rangle$ $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 moreover from $\langle \Psi \otimes \Psi_Q \otimes \Psi_P \vdash M \leftrightarrow K \rangle$
 have $\Psi \otimes \Psi_Q \otimes \Psi_P \vdash K \leftrightarrow M$
 by(*rule chanEqSym*)
 then have $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M$
 by(*blast intro: statEqEnt compositionSym Commutativity*)
 ultimately have $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu * xvec)(P' \parallel Q')$
 using $\langle A_P \ \#* \ \Psi \rangle$ $\langle A_P \ \#* \ P \rangle$ $\langle A_P \ \#* \ Q \rangle$ $\langle A_Q \ \#* \ A_P \rangle$ $\langle A_Q \ \#* \ \Psi \rangle$ $\langle A_Q \ \#* \ P \rangle$
 $\langle A_Q \ \#* \ Q \rangle$ $\langle xvec \ \#* \ Q \rangle$ $\langle A_P \ \#* \ K \rangle$ $\langle A_Q \ \#* \ M \rangle$
 by(*intro Comm2*) (*assumption | simp*)+
 moreover have $(\Psi, P' \parallel Q', Q' \parallel P') \in Rel$ **by**(*rule C1*)
 then have $(\Psi, (\nu * xvec)(P' \parallel Q'), (\nu * xvec)(Q' \parallel P')) \in Rel$ **using** $\langle xvec \ \#* \ \Psi \rangle$ **by**(*rule C2*)
 ultimately show *?case by blast*
 next
 case(*cComm2 Ψ_P M xvec N Q' A_Q Ψ_Q K P' A_P*)
 note $\langle \Psi \otimes \Psi_Q \triangleright P \mapsto K \langle N \rangle \prec P' \rangle$ $\langle \text{extractFrame } P = \langle A_P, \Psi_P \rangle \rangle$
 $\langle \Psi \otimes \Psi_P \triangleright Q \mapsto M(\nu * xvec) \langle N \rangle \prec Q' \rangle$ $\langle \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle \rangle$
 moreover from $\langle \Psi \otimes \Psi_Q \otimes \Psi_P \vdash M \leftrightarrow K \rangle$
 have $\Psi \otimes \Psi_Q \otimes \Psi_P \vdash K \leftrightarrow M$
 by(*rule chanEqSym*)
 then have $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash K \leftrightarrow M$
 by(*blast intro: statEqEnt compositionSym Commutativity*)
 ultimately have $\Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu * xvec)(P' \parallel Q')$
 using $\langle A_P \ \#* \ \Psi \rangle$ $\langle A_P \ \#* \ P \rangle$ $\langle A_P \ \#* \ Q \rangle$ $\langle A_Q \ \#* \ A_P \rangle$ $\langle A_Q \ \#* \ \Psi \rangle$ $\langle A_Q \ \#* \ P \rangle$
 $\langle A_Q \ \#* \ Q \rangle$ $\langle xvec \ \#* \ P \rangle$ $\langle A_P \ \#* \ K \rangle$ $\langle A_Q \ \#* \ M \rangle$
 by(*intro Comm1*) (*assumption | simp add: freshChainSym*)+
 moreover have $(\Psi, P' \parallel Q', Q' \parallel P') \in Rel$ **by**(*rule C1*)
 then have $(\Psi, (\nu * xvec)(P' \parallel Q'), (\nu * xvec)(Q' \parallel P')) \in Rel$ **using** $\langle xvec \ \#* \ \Psi \rangle$

```

Ψ › by(rule C2)
  ultimately show ?case by blast
next
  case(cBrMerge ΨP M N Q' AQ ΨQ P' AP)
  then have Ψ ▷ P ∥ Q ⟶i M(N) < P' ∥ Q'
    by(intro BrMerge) (assumption|simp)+
  then show ?case by(blast intro: C1)
next
  case(cBrComm1 ΨP M N Q' AQ ΨQ xvec P' AP)
  then have Ψ ▷ P ∥ Q ⟶i M(|ν*xvec|)(N) < P' ∥ Q'
    by(intro BrComm2) (assumption|simp)+
  then show ?case by(blast intro: C1)
next
  case(cBrComm2 ΨP M xvec N Q' AQ ΨQ P' AP)
  then have Ψ ▷ P ∥ Q ⟶i M(|ν*xvec|)(N) < P' ∥ Q'
    by(intro BrComm1) (assumption|simp)+
  then show ?case by(blast intro: C1)
qed
qed

lemma bangExtLeft:
  fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi

assumes guarded P
  and ∧Ψ' Q. (Ψ', Q, Q) ∈ Rel

shows Ψ ▷ !P ∼[Rel] P ∥ !P
  using assms
  by(auto simp add: simulation-def dest: Bang)

lemma bangExtRight:
  fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi

assumes C1: ∧Ψ' Q. (Ψ', Q, Q) ∈ Rel

shows Ψ ▷ P ∥ !P ∼[Rel] !P
proof –
  {
    fix α P'
    assume Ψ ▷ !P ⟶α < P'
    then have Ψ ▷ P ∥ !P ⟶α < P'
      apply –
      by(ind-cases Ψ ▷ !P ⟶α < P') (auto simp add: psi.inject)
    moreover have (Ψ, P', P') ∈ Rel by(rule C1)
    ultimately have ∃P''. Ψ ▷ P ∥ !P ⟶α < P'' ∧ (Ψ, P'', P') ∈ Rel
      by blast
  }

```



```

    then show ?thesis
      by(auto simp add: simulation-def)
qed

end

end

theory Bisim-Pres
  imports Bisimulation Sim-Pres
begin

  This file is a (heavily modified) variant of the theory Psi_Calculi.Bisim_Pres
  from [1].

  context env begin

  lemma bisimInputPres:
    fixes  $\Psi$     :: 'b
      and  $P$      :: ('a, 'b, 'c) psi
      and  $Q$      :: ('a, 'b, 'c) psi
      and  $M$      :: 'a
      and  $xvec$   :: name list
      and  $N$      :: 'a

    assumes  $\bigwedge Tvec. length\ xvec = length\ Tvec \implies \Psi \triangleright P[xvec ::= Tvec] \sim Q[xvec ::= Tvec]$ 

    shows  $\Psi \triangleright M(\lambda * xvec\ N).P \sim M(\lambda * xvec\ N).Q$ 
  proof -
    let ?X = {( $\Psi, M(\lambda * xvec\ N).P, M(\lambda * xvec\ N).Q$ ) |  $\Psi\ M\ xvec\ N\ P\ Q. \forall Tvec. length\ xvec = length\ Tvec \longrightarrow \Psi \triangleright P[xvec ::= Tvec] \sim Q[xvec ::= Tvec]$ }

    from assms have ( $\Psi, M(\lambda * xvec\ N).P, M(\lambda * xvec\ N).Q$ )  $\in$  ?X by blast
    then show ?thesis
      proof(coinduct rule: bisimCoinduct)
        case(cStatEq  $\Psi\ P\ Q$ )
          then show ?case by auto
        next
          case(cSim  $\Psi\ P\ Q$ )
            then show ?case by(blast intro: inputPres)
          next
            case(cExt  $\Psi\ P\ Q\ \Psi'$ )
              then show ?case by(blast dest: bisimE)
            next
              case(cSym  $\Psi\ P\ Q$ )
                then show ?case by(blast dest: bisimE)
            qed
          qed
    qed

  lemma bisimOutputPres:
    fixes  $\Psi$  :: 'b

```

```

and P :: ('a, 'b, 'c) psi
and Q :: ('a, 'b, 'c) psi
and M :: 'a
and N :: 'a

assumes Ψ ▷ P ~ Q

shows Ψ ▷ M⟨N⟩.P ~ M⟨N⟩.Q
proof -
  let ?X = {(Ψ, M⟨N⟩.P, M⟨N⟩.Q) | Ψ M N P Q. Ψ ▷ P ~ Q}
  from ⟨Ψ ▷ P ~ Q⟩ have (Ψ, M⟨N⟩.P, M⟨N⟩.Q) ∈ ?X by auto
  then show ?thesis
    by(coinduct rule: bisimCoinduct, auto) (blast intro: outputPres dest: bisimE)+
qed

lemma bisimCasePres:
  fixes Ψ :: 'b
    and CsP :: ('c × ('a, 'b, 'c) psi) list
    and CsQ :: ('c × ('a, 'b, 'c) psi) list

assumes ∧φ P. (φ, P) ∈ set CsP ⇒ ∃ Q. (φ, Q) ∈ set CsQ ∧ guarded Q ∧ Ψ
▷ P ~ Q
  and ∧φ Q. (φ, Q) ∈ set CsQ ⇒ ∃ P. (φ, P) ∈ set CsP ∧ guarded P ∧ Ψ ▷
P ~ Q

shows Ψ ▷ Cases CsP ~ Cases CsQ
proof -
  let ?X = {(Ψ, Cases CsP, Cases CsQ) | Ψ CsP CsQ. (∀φ P. (φ, P) ∈ set CsP
→ (∃ Q. (φ, Q) ∈ set CsQ ∧ guarded Q ∧ Ψ ▷ P ~ Q)) ∧
(∀φ Q. (φ, Q) ∈ set CsQ → (∃ P. (φ,
P) ∈ set CsP ∧ guarded P ∧ Ψ ▷ P ~ Q))}
  from assms have (Ψ, Cases CsP, Cases CsQ) ∈ ?X by auto
  then show ?thesis
    proof(coinduct rule: bisimCoinduct)
      case(cStatEq Ψ P Q)
        then show ?case by auto
    next
      case(cSim Ψ CasesP CasesQ)
        then obtain CsP CsQ where C1: ∧φ Q. (φ, Q) ∈ set CsQ ⇒ ∃ P. (φ, P)
∈ set CsP ∧ guarded P ∧ Ψ ▷ P ~ Q
        and A: CasesP = Cases CsP and B: CasesQ = Cases CsQ
        by auto
        note C1
        moreover have ∧Ψ P Q. Ψ ▷ P ~ Q ⇒ Ψ ▷ P ~→[bisim] Q by(rule bisimE)
        moreover have bisim ⊆ ?X ∪ bisim by blast
        ultimately have Ψ ▷ Cases CsP ~→[(?X ∪ bisim)] Cases CsQ
        by(rule casePres)
        then show ?case using A B by blast
    next

```

```

    case(cExt  $\Psi$   $P$   $Q$ )
    then show ?case by(blast dest: bisimE)
  next
    case(cSym  $\Psi$   $P$   $Q$ )
    then show ?case by(blast dest: bisimE)
  qed
qed

lemma bisimResPres:
  fixes  $\Psi$  :: 'b
  and  $P$  :: ('a, 'b, 'c) psi
  and  $Q$  :: ('a, 'b, 'c) psi
  and  $x$  :: name

assumes  $\Psi \triangleright P \sim Q$ 
  and  $x \# \Psi$ 

shows  $\Psi \triangleright (\nu x)P \sim (\nu x)Q$ 
proof -
  let ?X = {( $\Psi$ , ( $\nu *xvec$ ) $P$ , ( $\nu *xvec$ ) $Q$ ) |  $\Psi$   $xvec$   $P$   $Q$ .  $\Psi \triangleright P \sim Q \wedge xvec \#* \Psi$ }
  have eqvt ?X using bisimClosed pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]
    by(auto simp add: eqvts eqvt-def) blast
  have ( $\Psi$ , ( $\nu x$ ) $P$ , ( $\nu x$ ) $Q$ )  $\in$  ?X
  proof -
    from  $\langle x \# \Psi \rangle$  have  $[x] \#* \Psi$ 
      by auto
    with  $\langle \Psi \triangleright P \sim Q \rangle$  show ?thesis
      by (smt mem-Collect-eq resChain.base resChain.step)
  qed
  then show ?thesis
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq  $\Psi$   $xP$   $xQ$ )
    from  $\langle (\Psi, xP, xQ) \in ?X \rangle$  obtain  $xvec$   $P$   $Q$  where  $\Psi \triangleright P \sim Q$  and  $xvec \#*$ 
 $\Psi$  and  $xP = (\nu *xvec)P$  and  $xQ = (\nu *xvec)Q$ 
    by auto
    moreover from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $PeqQ$ : insertAssertion(extractFrame  $P$ )
 $\Psi \simeq_F$  insertAssertion(extractFrame  $Q$ )  $\Psi$ 
    by(rule bisimE)
    ultimately show ?case by(auto intro: frameResChainPres)
  next
    case(cSim  $\Psi$   $xP$   $xQ$ )
    from  $\langle (\Psi, xP, xQ) \in ?X \rangle$  obtain  $xvec$   $P$   $Q$  where  $\Psi \triangleright P \sim Q$  and  $xvec \#*$ 
 $\Psi$  and  $xP = (\nu *xvec)P$  and  $xQ = (\nu *xvec)Q$ 
    by auto
    from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \triangleright P \rightsquigarrow[bisim] Q$  by(rule bisimE)
    note  $\langle eqvt ?X \rangle$ 
    then have eqvt(?X  $\cup$  bisim) by auto
    from  $\langle xvec \#* \Psi \rangle$ 
    have  $\Psi \triangleright (\nu *xvec)P \rightsquigarrow[(?X \cup bisim)] (\nu *xvec)Q$ 

```

```

proof(induct xvec)
  case Nil
  have  $\Psi \triangleright (\nu^*[])P \rightsquigarrow[\text{bisim}] (\nu^*[])Q$  using  $\langle \Psi \triangleright P \sim Q \rangle$ 
    unfolding resChain.simps
    by(rule bisimE)
  then show ?case by(rule monotonic) auto
next
  case(Cons x xvec)
  then have  $x \# \Psi$  and  $xvec \#* \Psi$ 
    by auto
  from  $\langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu^*xvec)P \rightsquigarrow[(?X \cup \text{bisim})] (\nu^*xvec)Q$ 
    by(rule Cons)
  moreover note  $\langle \text{eqvt}(?X \cup \text{bisim}) \rangle$ 
  moreover note  $\langle x \# \Psi \rangle$ 
  moreover have  $?X \cup \text{bisim} \subseteq ?X \cup \text{bisim}$  by auto
  moreover have  $\bigwedge \Psi P Q xvec. [(\Psi, P, Q) \in ?X \cup \text{bisim}; xvec \#* \Psi] \implies (\Psi, (\nu^*xvec)P, (\nu^*xvec)Q) \in ?X \cup \text{bisim}$ 
    by (smt (verit, ccfv-threshold) Un-iff freshChainAppend mem-Collect-eq prod.inject resChainAppend)
  ultimately have  $\Psi \triangleright (\nu x)((\nu^*xvec)P) \rightsquigarrow[(?X \cup \text{bisim})] (\nu x)((\nu^*xvec)Q)$ 
    by(rule resPres)
  then show ?case unfolding resChain.simps by –
next
qed
with  $\langle xP = (\nu^*xvec)P \rangle$   $\langle xQ = (\nu^*xvec)Q \rangle$  show ?case
  by simp
next
  case(cExt  $\Psi$  xP xQ  $\Psi'$ )
  from  $\langle (\Psi, xP, xQ) \in ?X \rangle$  obtain  $xvec P Q$  where  $\Psi \triangleright P \sim Q$  and  $xvec \#* \Psi$ 
and  $xpe: xP = (\nu^*xvec)P$  and  $xqe: xQ = (\nu^*xvec)Q$ 
    by auto
  obtain  $p::\text{name prm}$  where  $(p.xvec) \#* \Psi$  and  $(p.xvec) \#* \Psi'$  and  $(p.xvec) \#* P$ 
and  $(p.xvec) \#* Q$  and  $(p.xvec) \#* xvec$  and distinctPerm p and  $set p \subseteq (set xvec) \times (set (p.xvec))$ 
    by(rule name-list-avoiding[where c=( $\Psi, \Psi', P, Q, xvec$ )] auto)
  from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \otimes (p \cdot \Psi') \triangleright P \sim Q$ 
    by(rule bisimE)
  moreover have  $xvec \#* (\Psi \otimes (p \cdot \Psi'))$  using  $\langle xvec \#* \Psi \rangle$   $\langle (p \cdot xvec) \#* \Psi' \rangle$ 
 $\langle \text{distinctPerm } p \rangle$ 
    apply(intro freshCompChain)
    apply assumption
    apply(subst perm-bool[where pi=p, symmetric])
    by(simp add: eqts)
  ultimately have  $(\Psi \otimes (p \cdot \Psi'), (\nu^*xvec)P, (\nu^*xvec)Q) \in ?X$ 
    by auto
  then have  $(p \cdot (\Psi \otimes (p \cdot \Psi')), (\nu^*xvec)P, (\nu^*xvec)Q) \in ?X$  using  $\langle \text{eqvt } ?X \rangle$ 
    by(intro eqvtI)
  then have  $(\Psi \otimes \Psi', (\nu^*(p.xvec))(p \cdot P), (\nu^*(p.xvec))(p \cdot Q)) \in ?X$  using  $\langle \text{distinctPerm } p \rangle$ 
 $\langle set p \subseteq (set xvec) \times (set (p.xvec)) \rangle$   $\langle xvec \#* \Psi \rangle$   $\langle (p \cdot xvec) \#* \Psi' \rangle$ 

```

```

    by(simp add: eqts)
  then have  $(\Psi \otimes \Psi', (\nu * xvec) P, (\nu * xvec) Q) \in ?X$  using  $\langle (p \cdot xvec) \#* Q \rangle \langle (p \cdot xvec) \#* P \rangle \langle set\ p \subseteq set\ xvec \times set\ (p \cdot xvec) \rangle \langle distinctPerm\ p \rangle$ 
    by(subst (1 2) resChainAlpha[where p=p]) auto
  then show ?case unfolding xpe xqe
    by blast
next
case(cSym  $\Psi\ P\ Q$ )
then show ?case
  by(blast dest: bisimE)
qed
qed

```

lemma bisimResChainPres:

```

fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c)\ psi$ 
and  $Q :: ('a, 'b, 'c)\ psi$ 
and  $xvec :: name\ list$ 

```

```

assumes  $\Psi \triangleright P \sim Q$ 
and  $xvec \#* \Psi$ 

```

```

shows  $\Psi \triangleright (\nu * xvec) P \sim (\nu * xvec) Q$ 
using assms
by(induct xvec) (auto intro: bisimResPres)

```

lemma bisimParPresAux:

```

fixes  $\Psi :: 'b$ 
and  $\Psi_R :: 'b$ 
and  $P :: ('a, 'b, 'c)\ psi$ 
and  $Q :: ('a, 'b, 'c)\ psi$ 
and  $R :: ('a, 'b, 'c)\ psi$ 
and  $A_R :: name\ list$ 

```

```

assumes  $\Psi \otimes \Psi_R \triangleright P \sim Q$ 
and  $FrR: extractFrame\ R = \langle A_R, \Psi_R \rangle$ 
and  $A_R \#* \Psi$ 
and  $A_R \#* P$ 
and  $A_R \#* Q$ 

```

```

shows  $\Psi \triangleright P \parallel R \sim Q \parallel R$ 

```

proof -

```

let ?X =  $\{(\Psi, (\nu * xvec)(P \parallel R), (\nu * xvec)(Q \parallel R)) \mid xvec\ \Psi\ P\ Q\ R.\ xvec\ \#* \Psi \wedge (\forall A_R\ \Psi_R.\ (extractFrame\ R = \langle A_R, \Psi_R \rangle \wedge A_R \#* \Psi \wedge A_R \#* P \wedge A_R \#* Q) \longrightarrow \Psi \otimes \Psi_R \triangleright P \sim Q)\}$ 

```

```

{
  fix xvec :: name list
  and  $\Psi :: 'b$ 

```

```

and  $P$   :: ('a, 'b, 'c) psi
and  $Q$   :: ('a, 'b, 'c) psi
and  $R$   :: ('a, 'b, 'c) psi

assume  $xvec \#* \Psi$ 
and  $\bigwedge A_R \Psi_R. \llbracket extractFrame R = \langle A_R, \Psi_R \rangle; A_R \#* \Psi; A_R \#* P; A_R \#* Q \rrbracket$ 
 $\implies \Psi \otimes \Psi_R \triangleright P \sim Q$ 

then have  $(\Psi, (\nu*xvec)(P \parallel R), (\nu*xvec)(Q \parallel R)) \in ?X$ 
by auto blast
}

note  $XI = this$ 
{
fix  $xvec :: name\ list$ 
and  $\Psi$   :: 'b
and  $P$   :: ('a, 'b, 'c) psi
and  $Q$   :: ('a, 'b, 'c) psi
and  $R$   :: ('a, 'b, 'c) psi
and  $C$   :: 'd::fs-name

assume  $xvec \#* \Psi$ 
and  $A: \bigwedge A_R \Psi_R. \llbracket extractFrame R = \langle A_R, \Psi_R \rangle; A_R \#* \Psi; A_R \#* P; A_R \#* Q; A_R \#* C \rrbracket \implies \Psi \otimes \Psi_R \triangleright P \sim Q$ 

from  $\langle xvec \#* \Psi \rangle$  have  $(\Psi, (\nu*xvec)(P \parallel R), (\nu*xvec)(Q \parallel R)) \in ?X$ 
proof(rule XI)
fix  $A_R \Psi_R$ 
assume  $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ 
obtain  $p::name\ prm$  where  $(p \cdot A_R) \#* \Psi$  and  $(p \cdot A_R) \#* P$  and  $(p \cdot A_R) \#* Q$  and  $(p \cdot A_R) \#* R$  and  $(p \cdot A_R) \#* C$ 
and  $(p \cdot A_R) \#* \Psi_R$  and  $S: (set\ p) \subseteq (set\ A_R) \times (set(p \cdot A_R))$  and
distinctPerm p
by(rule name-list-avoiding[where  $c=(\Psi, P, Q, R, \Psi_R, C)$ ]) auto
from  $FrR \langle (p \cdot A_R) \#* \Psi_R \rangle S$  have  $extractFrame R = \langle (p \cdot A_R), p \cdot \Psi_R \rangle$ 
by(simp add: frameChainAlpha')

moreover assume  $A_R \#* \Psi$ 
then have  $(p \cdot A_R) \#* (p \cdot \Psi)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle A_R \#* \Psi \rangle \langle (p \cdot A_R) \#* \Psi \rangle S$  have  $(p \cdot A_R) \#* \Psi$  by simp
moreover assume  $A_R \#* P$ 
then have  $(p \cdot A_R) \#* (p \cdot P)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle A_R \#* P \rangle \langle (p \cdot A_R) \#* P \rangle S$  have  $(p \cdot A_R) \#* P$  by simp
moreover assume  $A_R \#* Q$ 
then have  $(p \cdot A_R) \#* (p \cdot Q)$  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle A_R \#* Q \rangle \langle (p \cdot A_R) \#* Q \rangle S$  have  $(p \cdot A_R) \#* Q$  by simp

```

```

ultimately have  $\Psi \otimes (p \cdot \Psi_R) \triangleright P \sim Q$  using  $\langle (p \cdot A_R) \#^* C \rangle A$  by blast
then have  $(p \cdot (\Psi \otimes (p \cdot \Psi_R))) \triangleright (p \cdot P) \sim (p \cdot Q)$  by (rule bisimClosed)
with  $\langle A_R \#^* \Psi \rangle \langle (p \cdot A_R) \#^* \Psi \rangle \langle A_R \#^* P \rangle \langle (p \cdot A_R) \#^* P \rangle \langle A_R \#^* Q \rangle \langle (p \cdot A_R) \#^* Q \rangle S$  (distinctPerm p)
show  $\Psi \otimes \Psi_R \triangleright P \sim Q$  by (simp add: eqvts)
qed
}
note  $XI' = this$ 

have eqvt ?X
unfolding eqvt-def
proof
fix  $x$ 
assume  $x \in ?X$ 
then obtain  $xvec \Psi P Q R$  where 1:  $x = (\Psi, (\nu*xvec)P \parallel R, (\nu*xvec)Q \parallel R)$  and 2:  $xvec \#^* \Psi$  and
3:  $\forall A_R \Psi_R. extractFrame R = \langle A_R, \Psi_R \rangle \wedge A_R \#^* \Psi \wedge A_R \#^* P \wedge A_R \#^* Q \longrightarrow \Psi \otimes \Psi_R \triangleright P \sim Q$ 
by blast
show  $\forall p::(name \times name) list. p \cdot x \in ?X$ 
proof
fix  $p :: (name \times name) list$ 
from 1 have  $p \cdot x = (p \cdot \Psi, (\nu*p \cdot xvec)(p \cdot P) \parallel (p \cdot R), (\nu*p \cdot xvec)(p \cdot Q) \parallel (p \cdot R))$ 
by (simp add: eqvts)
moreover from 2 have  $(p \cdot xvec) \#^* (p \cdot \Psi)$ 
by (simp add: fresh-star-bij(2))
moreover have  $\forall A_R \Psi_R. extractFrame (p \cdot R) = \langle A_R, \Psi_R \rangle \wedge A_R \#^* (p \cdot \Psi) \wedge A_R \#^* (p \cdot P) \wedge A_R \#^* (p \cdot Q) \longrightarrow (p \cdot \Psi) \otimes \Psi_R \triangleright (p \cdot P) \sim (p \cdot Q)$ 
proof (rule allI, rule allI, rule impI, (erule conjE)+)
fix  $A_R \Psi_R$ 
assume  $exF: extractFrame (p \cdot R) = \langle A_R, \Psi_R \rangle$  and  $freshPsi: A_R \#^* (p \cdot \Psi)$  and  $freshP: A_R \#^* (p \cdot P)$  and  $freshQ: A_R \#^* (p \cdot Q)$ 
from  $exF$  have  $extractFrame R = \langle rev p \cdot A_R, rev p \cdot \Psi_R \rangle$ 
by (metis Chain.simps(5) extractFrameEqvt(1) frame.perm(1) frameResChainEqvt)
moreover from  $freshPsi$  have  $(rev p \cdot A_R) \#^* \Psi$ 
by (metis fresh-star-bij(2) perm-pi-simp(1))
moreover from  $freshP$  have  $(rev p \cdot A_R) \#^* P$ 
by (metis fresh-star-bij(2) perm-pi-simp(1))
moreover from  $freshQ$  have  $(rev p \cdot A_R) \#^* Q$ 
by (metis fresh-star-bij(2) perm-pi-simp(1))
ultimately show  $(p \cdot \Psi) \otimes \Psi_R \triangleright p \cdot P \sim p \cdot Q$ 
using 3 by (metis (no-types, opaque-lifting) bisimClosed perm-pi-simp(2) statEqvt')
qed
ultimately show  $p \cdot x \in ?X$ 
by blast
qed
qed

```

moreover have *Res*: $\bigwedge \Psi P Q x. \llbracket (\Psi, P, Q) \in ?X \cup \text{bisim}; x \# \Psi \rrbracket \implies (\Psi, (\nu x)P, (\nu x)Q) \in ?X \cup \text{bisim}$

proof –

fix $\Psi P Q x$

assume $(\Psi, P, Q) \in ?X \cup \text{bisim}$ **and** $(x::\text{name}) \# \Psi$

show $(\Psi, (\nu x)P, (\nu x)Q) \in ?X \cup \text{bisim}$

proof(*cases* $(\Psi, P, Q) \in ?X$)

assume $(\Psi, P, Q) \in ?X$

then obtain *xvec* $P' Q' R$ **where** $P = (\nu^*x\text{vec})P' \parallel R$ **and** $Q = (\nu^*x\text{vec})Q' \parallel R$ **and** *xvec* $\#^* \Psi$

and $\forall A_R \Psi_R. \text{extractFrame } R = \langle A_R, \Psi_R \rangle \wedge A_R \#^* \Psi \wedge A_R \#^* P' \wedge A_R \#^* Q' \longrightarrow \Psi \otimes \Psi_R \triangleright P' \sim Q'$

by *blast*

moreover have $(\nu x)((\nu^*x\text{vec})P' \parallel R) = (\nu^*(x \# \text{xvec}))P' \parallel R$

by *simp*

moreover have $(\nu x)((\nu^*x\text{vec})Q' \parallel R) = (\nu^*(x \# \text{xvec}))Q' \parallel R$

by *simp*

moreover from $\langle x \# \Psi \rangle \langle \text{xvec} \#^* \Psi \rangle$ **have** $(x \# \text{xvec}) \#^* \Psi$

by *simp*

ultimately have $(\Psi, (\nu x)P, (\nu x)Q) \in ?X$

by *blast*

then show *?thesis* **by** *simp*

next

assume $\neg(\Psi, P, Q) \in ?X$

with $\langle (\Psi, P, Q) \in ?X \cup \text{bisim} \rangle$ **have** $\Psi \triangleright P \sim Q$

by *blast*

then have $\Psi \triangleright (\nu x)P \sim (\nu x)Q$ **using** $\langle x \# \Psi \rangle$

by(*rule bisimResPres*)

then show *?thesis*

by *simp*

qed

qed

have *ResChain*: $\bigwedge \Psi P Q \text{xvec}. \llbracket (\Psi, P, Q) \in ?X \cup \text{bisim}; \text{xvec} \#^* \Psi \rrbracket \implies (\Psi, (\nu^*\text{xvec})P, (\nu^*\text{xvec})Q) \in ?X \cup \text{bisim}$

proof –

fix $\Psi P Q$

and *xvec::name list*

assume $(\Psi, P, Q) \in ?X \cup \text{bisim}$

and *xvec* $\#^* \Psi$

then show $(\Psi, (\nu^*\text{xvec})P, (\nu^*\text{xvec})Q) \in ?X \cup \text{bisim}$

proof(*induct xvec*)

case *Nil* **then show** *?case* **by** *simp*

next

case(*Cons x xvec*)

then have $(\Psi, (\nu^*\text{xvec})P, (\nu^*\text{xvec})Q) \in ?X \cup \text{bisim}$

by *simp*

moreover have $x \# \Psi$ **using** *Cons* **by** *simp*

ultimately show *?case* **unfolding** *resChain.simps*


```

    by(rule Res)
  qed
  qed
  have  $(\Psi, P \parallel R, Q \parallel R) \in ?X$ 
  proof -
    {
      fix  $A_{R'} :: \text{name list}$ 
      and  $\Psi_{R'} :: 'b$ 

      assume  $FrR'$ :  $\text{extractFrame } R = \langle A_{R'}, \Psi_{R'} \rangle$ 
      and  $A_{R'} \#* \Psi$ 
      and  $A_{R'} \#* P$ 
      and  $A_{R'} \#* Q$ 

      obtain  $p$  where  $(p \cdot A_{R'}) \#* A_R$  and  $(p \cdot A_{R'}) \#* \Psi_{R'}$  and  $(p \cdot A_{R'}) \#* \Psi$ 
    and  $(p \cdot A_{R'}) \#* P$  and  $(p \cdot A_{R'}) \#* Q$ 
      and  $Sp$ :  $(\text{set } p) \subseteq (\text{set } A_{R'}) \times (\text{set}(p \cdot A_{R'}))$  and  $\text{distinctPerm } p$ 
      by(rule name-list-avoiding[where  $c=(A_R, \Psi, \Psi_{R'}, P, Q)$ ]) auto

      from  $FrR'$   $\langle (p \cdot A_{R'}) \#* \Psi_{R'} \rangle Sp$  have  $\text{extractFrame } R = \langle (p \cdot A_{R'}), p \cdot \Psi_{R'} \rangle$ 
      by(simp add: frameChainAlpha eqvts)
      with  $FrR$   $\langle (p \cdot A_{R'}) \#* A_R \rangle$  obtain  $q :: \text{name prm}$ 
      where  $Sq$ :  $\text{set } q \subseteq \text{set}(p \cdot A_{R'}) \times \text{set } A_R$  and  $\text{distinctPerm } q$  and  $\Psi_R = q \cdot p \cdot \Psi_{R'}$ 
      by(force elim: frameChainEq)

      from  $\langle \Psi \otimes \Psi_R \triangleright P \sim Q \rangle \langle \Psi_R = q \cdot p \cdot \Psi_{R'} \rangle$  have  $\Psi \otimes (q \cdot p \cdot \Psi_{R'}) \triangleright P \sim Q$  by simp
      then have  $(q \cdot (\Psi \otimes (q \cdot p \cdot \Psi_{R'}))) \triangleright (q \cdot P) \sim (q \cdot Q)$  by(rule bisimClosed)
      with  $Sq$   $\langle A_R \#* \Psi \rangle \langle (p \cdot A_{R'}) \#* \Psi \rangle \langle A_R \#* P \rangle \langle (p \cdot A_{R'}) \#* P \rangle \langle A_R \#* Q \rangle \langle (p \cdot A_{R'}) \#* Q \rangle \langle \text{distinctPerm } q \rangle$ 
      have  $\Psi \otimes (p \cdot \Psi_{R'}) \triangleright P \sim Q$  by(simp add: eqvts)
      then have  $(p \cdot (\Psi \otimes (p \cdot \Psi_{R'}))) \triangleright (p \cdot P) \sim (p \cdot Q)$  by(rule bisimClosed)
      with  $Sp$   $\langle A_{R'} \#* \Psi \rangle \langle (p \cdot A_{R'}) \#* \Psi \rangle \langle A_{R'} \#* P \rangle \langle (p \cdot A_{R'}) \#* P \rangle \langle A_{R'} \#* Q \rangle \langle (p \cdot A_{R'}) \#* Q \rangle \langle \text{distinctPerm } p \rangle$ 
      have  $\Psi \otimes \Psi_{R'} \triangleright P \sim Q$  by(simp add: eqvts)
    }
    then show ?thesis
      apply clarsimp
      apply(rule exI[where  $x=[]$ ])
      by auto blast
  qed
  then show ?thesis
  proof(coinduct rule: bisimCoinduct)
    case(cStatEq  $\Psi PR QR$ )
    from  $\langle \Psi, PR, QR \rangle \in ?X$ 
    obtain  $xvec$   $P$   $Q$   $R$  where  $PFrR$ :  $PR = (\nu * xvec)(P \parallel R)$  and  $QFrR$ :  $QR = (\nu * xvec)(Q \parallel R)$ 

```

and $xvec \#* \Psi$ **and** $*$: $\forall A_R \Psi_R. extractFrame R = \langle A_R, \Psi_R \rangle \wedge A_R \#* \Psi \wedge A_R \#* P \wedge A_R \#* Q \longrightarrow \Psi \otimes \Psi_R \triangleright P \sim Q$
by *blast*
obtain $A_R \Psi_R$ **where** $FrR: extractFrame R = \langle A_R, \Psi_R \rangle$ **and** *fresh*: $A_R \#* (\Psi, P, Q, R)$
using *freshFrame by metis*
from *fresh* **have** $A_R \#* \Psi$ **and** $A_R \#* P$ **and** $A_R \#* Q$ **and** $A_R \#* R$
by *auto*
with FrR **have** $PSimQ: \Psi \otimes \Psi_R \triangleright P \sim Q$
using $*$ **by** *blast*

obtain $A_P \Psi_P$ **where** $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#* \Psi$ **and** $A_P \#* A_R$ **and** $A_P \#* \Psi_R$
by (*rule freshFrame[where C=(Ψ, A_R, Ψ_R)*]) *auto*
obtain $A_Q \Psi_Q$ **where** $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** $A_Q \#* \Psi$ **and** $A_Q \#* A_R$ **and** $A_Q \#* \Psi_R$
by (*rule freshFrame[where C=(Ψ, A_R, Ψ_R)*]) *auto*
from $\langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle A_P \#* A_R \rangle \langle A_Q \#* A_R \rangle$ $FrP FrQ$ **have** $A_R \#* \Psi_P$ **and** $A_R \#* \Psi_Q$
by (*force dest: extractFrameFreshChain*) $+$

have $\langle (A_P @ A_R), \Psi \otimes \Psi_P \otimes \Psi_R \rangle \simeq_F \langle (A_Q @ A_R), \Psi \otimes \Psi_Q \otimes \Psi_R \rangle$
proof –
have $\langle A_P, \Psi \otimes \Psi_P \otimes \Psi_R \rangle \simeq_F \langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle$
by (*metis frameResChainPres frameNilStatEq Associativity Commutativity AssertionStatEqTrans Composition*)
moreover from $PSimQ$ **have** $insertAssertion(extractFrame P) (\Psi \otimes \Psi_R) \simeq_F insertAssertion(extractFrame Q) (\Psi \otimes \Psi_R)$
by (*rule bisimE*)
with $FrP FrQ$ *freshCompChain* $\langle A_P \#* \Psi \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* \Psi \rangle \langle A_Q \#* \Psi_R \rangle$ **have** $\langle A_P, (\Psi \otimes \Psi_R) \otimes \Psi_P \rangle \simeq_F \langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle$
by *auto*
moreover have $\langle A_Q, (\Psi \otimes \Psi_R) \otimes \Psi_Q \rangle \simeq_F \langle A_Q, \Psi \otimes \Psi_Q \otimes \Psi_R \rangle$
by (*metis frameResChainPres frameNilStatEq Associativity Commutativity AssertionStatEqTrans Composition*)
ultimately have $\langle A_P, \Psi \otimes \Psi_P \otimes \Psi_R \rangle \simeq_F \langle A_Q, \Psi \otimes \Psi_Q \otimes \Psi_R \rangle$
by (*blast intro: FrameStatEqTrans*)
then have $\langle (A_R @ A_P), \Psi \otimes \Psi_P \otimes \Psi_R \rangle \simeq_F \langle (A_R @ A_Q), \Psi \otimes \Psi_Q \otimes \Psi_R \rangle$
by (*drule-tac frameResChainPres*) (*simp add: frameChainAppend*)
then show *?thesis*
apply (*simp add: frameChainAppend*)
by (*metis frameResChainComm FrameStatEqTrans*)

qed
moreover from $\langle A_P \#* \Psi \rangle \langle A_R \#* \Psi \rangle$ **have** $(A_P @ A_R) \#* \Psi$ **by** *simp*
moreover from $\langle A_Q \#* \Psi \rangle \langle A_R \#* \Psi \rangle$ **have** $(A_Q @ A_R) \#* \Psi$ **by** *simp*
ultimately have $P FrR QR: insertAssertion(extractFrame(P \parallel R)) \Psi \simeq_F insertAssertion(extractFrame(Q \parallel R)) \Psi$
using $FrP FrQ FrR$ $\langle A_P \#* A_R \rangle \langle A_P \#* \Psi_R \rangle \langle A_Q \#* A_R \rangle \langle A_Q \#* \Psi_R \rangle \langle A_R \#* \Psi_P \rangle \langle A_R \#* \Psi_Q \rangle$

```

by simp

from ⟨xvec #* Ψ⟩ have insertAssertion (extractFrame(⟨ν*xvec⟩P || R)) Ψ ≈F
(⟨ν*xvec⟩(insertAssertion (extractFrame(P || R)) Ψ))
  by(rule insertAssertionExtractFrameFresh)
moreover from PFrRQR have (⟨ν*xvec⟩(insertAssertion (extractFrame(P ||
R)) Ψ)) ≈F (⟨ν*xvec⟩(insertAssertion (extractFrame(Q || R)) Ψ))
  by(induct xvec) (auto intro: frameResPres)
moreover from ⟨xvec #* Ψ⟩ have (⟨ν*xvec⟩(insertAssertion (extractFrame(Q
|| R)) Ψ)) ≈F insertAssertion (extractFrame(⟨ν*xvec⟩Q || R)) Ψ
  by(rule FrameStatEqSym[OF insertAssertionExtractFrameFresh])
ultimately show ?case using PFrR QFrR
  by(blast intro: FrameStatEqTrans)
next
case(cSim Ψ PR QR)
{
  fix Ψ P Q R xvec
  assume ∧AR ΨR. [extractFrame R = ⟨AR, ΨR⟩; AR #* Ψ; AR #* P; AR #*
Q] ⇒ Ψ ⊗ ΨR ▷ P ~ Q
  moreover have eqvt bisim by simp
  moreover from ⟨eqvt ?X⟩ have eqvt(?X ∪ bisim) by auto
  moreover from bisimE(1) have ∧Ψ P Q. Ψ ▷ P ~ Q ⇒ insertAsser-
tion (extractFrame Q) Ψ ↦F insertAssertion (extractFrame P) Ψ by(simp add:
FrameStatEq-def)
  moreover note bisimE(2) bisimE(3)
  moreover
  {
    fix Ψ P Q AR ΨR R
    assume PSimQ: Ψ ⊗ ΨR ▷ P ~ Q
    and FrR: extractFrame R = ⟨AR, ΨR⟩
    and AR #* Ψ
    and AR #* P
    and AR #* Q
    then have (Ψ, P || R, Q || R) ∈ ?X
    proof -
      have P || R = (⟨ν*[]⟩)(P || R) by simp
      moreover have Q || R = (⟨ν*[]⟩)(Q || R) by simp
      moreover have ([]::name list) #* Ψ by simp
      moreover
      {
        fix AR' ΨR'

        assume FrR': extractFrame R = ⟨AR', ΨR'⟩
        and AR' #* Ψ
        and AR' #* P
        and AR' #* Q
        obtain p where (p · AR') #* AR
        and (p · AR') #* ΨR'
        and (p · AR') #* Ψ
      }
    }
  }
}

```

```

and  $\langle p \cdot A_R \rangle \#* P$ 
and  $\langle p \cdot A_R \rangle \#* Q$ 
and  $S: (set\ p) \subseteq (set\ A_R) \times (set\ (p \cdot A_R))$  and  $distinctPerm\ p$ 
by( $rule\ name-list-avoiding[where\ c=(A_R, \Psi, \Psi_R', P, Q)]$ )  $auto$ 

from  $\langle p \cdot A_R \rangle \#* \Psi_R' \rangle S$  have  $\langle A_R', \Psi_R \rangle = \langle p \cdot A_R', p \cdot \Psi_R \rangle$ 
by( $simp\ add: frameChainAlpha$ )

with  $FrR'$  have  $FrR''$ :  $extractFrame\ R = \langle p \cdot A_R', p \cdot \Psi_R \rangle$  by  $simp$ 
with  $FrR$   $\langle p \cdot A_R \rangle \#* A_R$ 
obtain  $q$  where  $p \cdot \Psi_R' = (q::name\ prm) \cdot \Psi_R$  and  $S': set\ q \subseteq (set\ A_R) \times set\ (p \cdot A_R)$  and  $distinctPerm\ q$ 
apply  $clarsimp$ 
apply( $drule\ sym$ )
apply  $simp$ 
by( $drule\ frameChainEq$ )  $auto$ 
from  $PSimQ$  have  $\langle q \cdot (\Psi \otimes \Psi_R) \rangle \triangleright \langle q \cdot P \rangle \sim \langle q \cdot Q \rangle$ 
by( $rule\ bisimClosed$ )
with  $\langle A_R \#* \Psi \rangle \langle A_R \#* P \rangle \langle A_R \#* Q \rangle \langle p \cdot A_R \rangle \#* \Psi \rangle \langle p \cdot A_R \rangle \#* P \rangle$ 
 $\langle p \cdot A_R \rangle \#* Q \rangle S'$ 
have  $\Psi \otimes (q \cdot \Psi_R) \triangleright P \sim Q$  by( $simp\ add: eqvts$ )
then have  $\langle p \cdot (\Psi \otimes (q \cdot \Psi_R)) \rangle \triangleright \langle p \cdot P \rangle \sim \langle p \cdot Q \rangle$  by( $rule\ bisimClosed$ )
with  $\langle A_R' \#* \Psi \rangle \langle A_R' \#* P \rangle \langle A_R' \#* Q \rangle \langle p \cdot A_R \rangle \#* \Psi \rangle \langle p \cdot A_R \rangle \#*$ 
 $P \rangle \langle p \cdot A_R \rangle \#* Q \rangle S$   $\langle distinctPerm\ p \rangle \langle p \cdot \Psi_R' \rangle = q \cdot \Psi_R$ 
have  $\Psi \otimes \Psi_R' \triangleright P \sim Q$ 
by( $drule-tac\ sym$ ) ( $simp\ add: eqvts$ )
}
ultimately show  $?thesis$ 
by  $blast$ 
qed
then have  $(\Psi, P \parallel R, Q \parallel R) \in ?X \cup bisim$ 
by  $simp$ 
}
moreover have  $\bigwedge \Psi\ P\ Q\ xvec. \llbracket (\Psi, P, Q) \in ?X \cup bisim; (xvec::name\ list) \#* \Psi \rrbracket \implies (\Psi, (\nu*xvec)P, (\nu*xvec)Q) \in ?X \cup bisim$ 
proof –
fix  $\Psi\ P\ Q\ xvec$ 
assume  $(\Psi, P, Q) \in ?X \cup bisim$ 
assume  $(xvec::name\ list) \#* \Psi$ 
then show  $(\Psi, (\nu*xvec)P, (\nu*xvec)Q) \in ?X \cup bisim$ 
proof( $induct\ xvec$ )
case  $Nil$ 
then show  $?case$  using  $\langle (\Psi, P, Q) \in ?X \cup bisim \rangle$  by  $simp$ 
next
case( $Cons\ x\ xvec$ )
then show  $?case$  by( $simp\ only: resChain.simps$ ) ( $rule\ Res, auto$ )
qed
qed
ultimately have  $\Psi \triangleright P \parallel R \rightsquigarrow[(?X \cup bisim)] Q \parallel R$  using  $statEqBisim$ 

```

```

    by(rule parPres)
    moreover assume (xvec::name list) #* Ψ
    moreover from ⟨eqvt ?X⟩ have eqvt(?X ∪ bisim) by auto
    ultimately have Ψ ▷ (ν*xvec)(P ∥ R) ∼[ (?X ∪ bisim)] (ν*xvec)(Q ∥ R)
using ResChain
    by(intro resChainPres)
  }
  with ⟨(Ψ, PR, QR) ∈ ?X⟩ show ?case by blast
next
  case(cExt Ψ PR QR Ψ')

  from ⟨(Ψ, PR, QR) ∈ ?X⟩
  obtain xvec P Q R AR ΨR where PFrR: PR = (ν*xvec)(P ∥ R) and QFrR:
QR = (ν*xvec)(Q ∥ R)
  and xvec #* Ψ and A: ∀ AR ΨR}. (extractFrame R = ⟨AR, ΨR⟩ ∧ AR #* Ψ ∧
AR #* P ∧ AR #* Q) → Ψ ⊗ ΨR ▷ P ∼ Q
  by auto

  obtain p where (p · xvec) #* Ψ
  and (p · xvec) #* P
  and (p · xvec) #* Q
  and (p · xvec) #* R
  and (p · xvec) #* Ψ'
  and S: (set p) ⊆ (set xvec) × (set(p · xvec)) and distinctPerm p
  by(rule name-list-avoiding[where c=(Ψ, P, Q, R, Ψ')]) auto

  from ⟨(p · xvec) #* P⟩ ⟨(p · xvec) #* R⟩ S have (ν*xvec)(P ∥ R) = (ν*(p ·
xvec))(p · (P ∥ R))
  by(subst resChainAlpha) auto
  then have PRAlpha: (ν*xvec)(P ∥ R) = (ν*(p · xvec))((p · P) ∥ (p · R))
  by(simp add: eqts)

  from ⟨(p · xvec) #* Q⟩ ⟨(p · xvec) #* R⟩ S have (ν*xvec)(Q ∥ R) = (ν*(p ·
xvec))(p · (Q ∥ R))
  by(subst resChainAlpha) auto
  then have QRAlpha: (ν*xvec)(Q ∥ R) = (ν*(p · xvec))((p · Q) ∥ (p · R))
  by(simp add: eqts)

  have (Ψ ⊗ Ψ', (ν*(p · xvec))((p · P) ∥ (p · R)), (ν*(p · xvec))((p · Q) ∥ (p ·
R))) ∈ ?X
  proof(rule XI'[where C2=(Ψ, (p · P), (p · Q), R, Ψ', xvec, p · xvec)])
    show (p · xvec) #* (Ψ ⊗ Ψ')
    using ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec) #* Ψ'⟩ by auto
  next
  fix AR ΨR
  assume FrR: extractFrame (p · R) = ⟨AR, ΨR⟩ and AR #* (Ψ ⊗ Ψ') and
AR #* (p · P) and AR #* (Ψ, p · P, p · Q, R, Ψ', xvec, p · xvec)
  then have AR #* Ψ and AR #* (p · Q)
  by simp-all

```

```

from FrR have  $(p \cdot (\text{extractFrame } (p \cdot R))) = (p \cdot \langle A_R, \Psi_R \rangle)$ 
  by simp
with  $\langle \text{distinctPerm } p \rangle$  have  $\text{extractFrame } R = \langle p \cdot A_R, p \cdot \Psi_R \rangle$ 
  by(simp add: eqvts)
moreover from  $\langle A_R \#* \Psi \rangle$  have  $(p \cdot A_R) \#* (p \cdot \Psi)$ 
  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle \text{xvec } \#* \Psi \rangle \langle (p \cdot \text{xvec}) \#* \Psi \rangle S$  have  $(p \cdot A_R) \#* \Psi$ 
  by simp
moreover from  $\langle A_R \#* (p \cdot P) \rangle$  have  $(p \cdot A_R) \#* (p \cdot p \cdot P)$ 
  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle \text{distinctPerm } p \rangle$  have  $(p \cdot A_R) \#* P$ 
  by simp
moreover from  $\langle A_R \#* (p \cdot Q) \rangle$  have  $(p \cdot A_R) \#* (p \cdot p \cdot Q)$ 
  by(simp add: pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst])
with  $\langle \text{distinctPerm } p \rangle$  have  $(p \cdot A_R) \#* Q$ 
  by simp
ultimately have  $\Psi \otimes (p \cdot \Psi_R) \triangleright P \sim Q$ 
  using A by blast
then have  $(\Psi \otimes (p \cdot \Psi_R)) \otimes (p \cdot \Psi') \triangleright P \sim Q$ 
  by(rule bisimE)
moreover have  $(\Psi \otimes (p \cdot \Psi_R)) \otimes (p \cdot \Psi') \simeq (\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R)$ 
  by(metis Associativity Commutativity Composition AssertionStatEqTrans
AssertionStatEqSym)
ultimately have  $(\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R) \triangleright P \sim Q$ 
  by(rule statEqBisim)
then have  $(p \cdot ((\Psi \otimes (p \cdot \Psi')) \otimes (p \cdot \Psi_R))) \triangleright (p \cdot P) \sim (p \cdot Q)$ 
  by(rule bisimClosed)
with  $\langle \text{distinctPerm } p \rangle \langle \text{xvec } \#* \Psi \rangle \langle (p \cdot \text{xvec}) \#* \Psi \rangle S$  show  $(\Psi \otimes \Psi') \otimes \Psi_R$ 
 $\triangleright (p \cdot P) \sim (p \cdot Q)$ 
  by(simp add: eqvts)
qed
with PFrR QFrR PRAlpha QRAlpha show ?case by simp
next
  case(cSym  $\Psi$  PR QR)
  then show ?case by(blast dest: bisimE)
qed
qed

```

lemma *bisimParPres*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{psi}$ 
  and  $R :: ('a, 'b, 'c) \text{psi}$ 

```

assumes $\Psi \triangleright P \sim Q$

shows $\Psi \triangleright P \parallel R \sim Q \parallel R$

proof –

obtain $A_R \Psi_R$ **where** $\text{extractFrame } R = \langle A_R, \Psi_R \rangle$ **and** $A_R \#* \Psi$ **and** $A_R \#* P$

```

and  $A_R \#^* Q$ 
  by(rule freshFrame[where  $C=(\Psi, P, Q)$ ]) auto
  moreover from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \otimes \Psi_R \triangleright P \sim Q$  by(rule bisimE)
  ultimately show ?thesis by(intro bisimParPresAux)
qed

end

end

theory Bisim-Struct-Cong
  imports Bisim-Pres Sim-Struct-Cong Psi-Calculi.Structural-Congruence
begin

  This file is a (heavily modified) variant of the theory Psi_Calculi.Bisim_Struct_Cong
  from [1].

  context env begin

  lemma bisimParComm:
    fixes  $\Psi :: 'b$ 
      and  $P :: ('a, 'b, 'c) \text{ psi}$ 
      and  $Q :: ('a, 'b, 'c) \text{ psi}$ 

  shows  $\Psi \triangleright P \parallel Q \sim Q \parallel P$ 
  proof -
    let  $?X = \{((\Psi::'b), (\nu^*xvec)((P::('a, 'b, 'c) \text{ psi}) \parallel Q), (\nu^*xvec)(Q \parallel P)) \mid xvec$ 
     $\Psi P Q. xvec \#^* \Psi\}$ 

    have eqvt ?X
      by(force simp add: eqvt-def pt-fresh-star-bij[OF pt-name-inst, OF at-name-inst]
      eqvts)

    have  $(\Psi, P \parallel Q, Q \parallel P) \in ?X$ 
      using resChain.base by(smt (verit) freshSets(1) mem-Collect-eq)
    then show ?thesis
    proof(coinduct rule: bisimWeakCoinduct)
      case(cStatEq  $\Psi PQ QP$ )
      from  $\langle (\Psi, PQ, QP) \in ?X \rangle$ 
      obtain  $xvec P Q$  where  $PFrQ: PQ = (\nu^*xvec)(P \parallel Q)$  and  $QFrP: QP =$ 
       $(\nu^*xvec)(Q \parallel P)$  and  $xvec \#^* \Psi$ 
      by auto

      obtain  $A_P \Psi_P$  where  $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$  and  $A_P \#^* \Psi$  and
       $A_P \#^* Q$ 
      by(rule freshFrame[where  $C=(\Psi, Q)$ ]) auto
      obtain  $A_Q \Psi_Q$  where  $FrQ: extractFrame Q = \langle A_Q, \Psi_Q \rangle$  and  $A_Q \#^* \Psi$  and
       $A_Q \#^* A_P$  and  $A_Q \#^* \Psi_P$ 
      by(rule freshFrame[where  $C=(\Psi, A_P, \Psi_P)$ ]) auto
      from  $FrQ \langle A_Q \#^* A_P \rangle \langle A_P \#^* Q \rangle$  have  $A_P \#^* \Psi_Q$  by(force dest: extractFrame-
      FreshChain)

```

```

have  $\langle (xvec @ A_P @ A_Q), \Psi \otimes \Psi_P \otimes \Psi_Q \rangle \simeq_F \langle (xvec @ A_Q @ A_P), \Psi \otimes \Psi_Q \otimes \Psi_P \rangle$ 
by (simp add: frameChainAppend)
      (metis frameResChainPres frameResChainComm frameNilStatEq compositionSym Associativity Commutativity FrameStatEqTrans)
with FrP FrQ PFrQ QFrP  $\langle A_P \#* \Psi_Q \rangle \langle A_Q \#* \Psi_P \rangle \langle A_Q \#* A_P \rangle \langle xvec \#* \Psi \rangle$ 
 $\langle A_P \#* \Psi \rangle \langle A_Q \#* \Psi \rangle$ 
show ?case by (auto simp add: frameChainAppend)
next
case (cSim  $\Psi$  PQ QP)
from  $\langle (\Psi, PQ, QP) \in ?X \rangle$ 
obtain xvec P Q where PFrQ:  $PQ = (\nu * xvec)(P \parallel Q)$  and QFrP:  $QP =$ 
 $(\nu * xvec)(Q \parallel P)$ 
and xvec  $\#* \Psi$ 
by auto
moreover have  $\Psi \triangleright (\nu * xvec)(P \parallel Q) \rightsquigarrow[?X] (\nu * xvec)(Q \parallel P)$ 
proof –
have  $\Psi \triangleright P \parallel Q \rightsquigarrow[?X] Q \parallel P$ 
proof –
note  $\langle eqvt ?X \rangle$ 
moreover have  $\bigwedge \Psi P Q. (\Psi, P \parallel Q, Q \parallel P) \in ?X$ 
using resChain.base by (smt (verit) freshSets(1) mem-Collect-eq)
moreover have  $\bigwedge \Psi P Q xvec. [(\Psi, P, Q) \in ?X; xvec \#* \Psi] \implies (\Psi,$ 
 $(\nu * xvec)P, (\nu * xvec)Q) \in ?X$ 
proof (induct xvec)
case Nil
then show ?case
by (smt (verit, ccfv-threshold) Pair-inject freshChainAppend mem-Collect-eq
resChainAppend)
next
case (Cons a xvec)
then show ?case
by blast
qed
ultimately show ?thesis by (rule simParComm)
qed
moreover note  $\langle eqvt ?X \rangle \langle xvec \#* \Psi \rangle$ 
moreover have  $\bigwedge \Psi P Q xvec. [(\Psi, P, Q) \in ?X; xvec \#* \Psi] \implies (\Psi,$ 
 $(\nu * xvec)P, (\nu * xvec)Q) \in ?X$ 
using resChainAppend[symmetric] freshChainAppend by blast
ultimately show ?thesis
by (rule resChainPres)
qed
ultimately show ?case by simp
next
case (cExt  $\Psi$  PQ QP  $\Psi'$ )
from  $\langle (\Psi, PQ, QP) \in ?X \rangle$ 
obtain xvec P Q where PFrQ:  $PQ = (\nu * xvec)(P \parallel Q)$  and QFrP:  $QP =$ 
 $(\nu * xvec)(Q \parallel P)$ 
and xvec  $\#* \Psi$ 

```



```

    by auto

  obtain p where (p · xvec) #* Ψ
    and (p · xvec) #* P
    and (p · xvec) #* Q
    and (p · xvec) #* Ψ'
    and S: (set p) ⊆ (set xvec) × (set(p · xvec)) and distinctPerm p
    by(rule name-list-avoiding[where c=(Ψ, P, Q, Ψ')]) auto

  from ⟨(p · xvec) #* P⟩ ⟨(p · xvec) #* Q⟩ S have (ν*xvec)(P || Q) = (ν*(p ·
xvec))(p · (P || Q))
    by(subst resChainAlpha) auto
  then have PQAlpha: (ν*xvec)(P || Q) = (ν*(p · xvec))((p · P) || (p · Q))
    by(simp add: eqts)

  from ⟨(p · xvec) #* P⟩ ⟨(p · xvec) #* Q⟩ S have (ν*xvec)(Q || P) = (ν*(p ·
xvec))(p · (Q || P))
    by(subst resChainAlpha) auto
  then have QPAlpha: (ν*xvec)(Q || P) = (ν*(p · xvec))((p · Q) || (p · P))
    by(simp add: eqts)

  from ⟨(p · xvec) #* Ψ⟩ ⟨(p · xvec) #* Ψ'⟩ have (Ψ ⊗ Ψ', (ν*(p · xvec))((p · P)
|| (p · Q)), (ν*(p · xvec))((p · Q) || (p · P))) ∈ ?X
    by auto
  with PFrQ QFrP PQAlpha QPAlpha show ?case by simp
next
  case(cSym Ψ PR QR)
  then show ?case by blast
qed
qed

inductive-set resCommRel :: ('b × ('a,'b,'c) psi × ('a,'b,'c) psi) set
  where
    resCommRel-refl : (Ψ,P,P) ∈ resCommRel
  | resCommRel-swap : [[a # Ψ; b # Ψ] ⇒ (Ψ,(νa)((νb)P),(νb)((νa)P)) ∈ resComm-
Rel
  | resCommRel-res : [(Ψ,P,Q) ∈ resCommRel; a # Ψ] ⇒ (Ψ,(νa)P,(νa)Q) ∈
resCommRel

lemma eqtResCommRel: eqt resCommRel
proof -
  {
    fix Ψ P Q
    and p::name prm
    assume (Ψ,P,Q) ∈ resCommRel
    then have (p·Ψ,p·P,p·Q) ∈ resCommRel
    proof(induct rule: resCommRel.inducts)
      case resCommRel-refl then show ?case by(rule resCommRel.intros)
    next
  }

```

```

    case(resCommRel-swap a Ψ b P)
  then have (p·a) # p·Ψ and (p·b) # p·Ψ
    apply –
    by(subst (asm) (1 2) perm-bool[where pi=p,symmetric], simp add: eqvts)+
  then show ?case unfolding eqvts
    by(rule resCommRel.intros)
next
case(resCommRel-res Ψ P Q a)
from ⟨a # Ψ⟩ have (p·a) # p·Ψ
  apply –
  by(subst (asm) perm-bool[where pi=p,symmetric], simp add: eqvts)
then show ?case using ⟨p·Ψ, p·P, p·Q⟩ ∈ resCommRel
  unfolding eqvts
  by(intro resCommRel.intros)
qed
}
then show ?thesis unfolding eqvt-def
  by auto
qed

```

```

lemma resCommRelStarRes:
  assumes (Ψ,P,Q) ∈ resCommRel*
    and a # Ψ
  shows (Ψ,(νa)P,(νa)Q) ∈ resCommRel*
  using assms
proof(induct rule: rel-trancl.induct)
  case r-into-rel-trancl then show ?case by(auto intro: resCommRel-res)
next
case(rel-trancl-into-rel-trancl Ψ P Q R)
  then show ?case
    by(auto dest: resCommRel-res intro: rel-trancl.intros)
qed

```

```

lemma resCommRelStarResChain:
  assumes (Ψ,P,Q) ∈ resCommRel*
    and xvec #* Ψ
  shows (Ψ,(ν*xvec)P,(ν*xvec)Q) ∈ resCommRel*
  using assms
  by(induct xvec) (auto simp add: resCommRelStarRes)

```

```

lemma length-induct1 [consumes 0, case-names Nil Cons]:
  assumes b: P []
    and s: ⋀ x yvec. [⋀ yvec. length xvec=length yvec ⇒ P yvec] ⇒ P(x#yvec)
  shows P xvec
proof(induct xvec rule: length-induct)
  case(1 xvec)
  then show ?case
  proof(cases xvec)
    case Nil then show ?thesis by(simp add: b)

```

```

next
  case(Cons y yvec)
  with  $\langle \forall ys. \text{length } ys < \text{length } xvec \longrightarrow P \ ys \rangle$  have  $\forall ys. \text{length } ys = \text{length } yvec \longrightarrow P \ ys$  by simp
  then show ?thesis unfolding Cons
  using s by auto
qed

```

lemma *oneStepPerm-in-rel*:

```

assumes  $xvec \#* \Psi$ 
shows  $(\Psi, (\nu * xvec) P, (\nu * (\text{rotate1 } xvec)) P) \in \text{resCommRel}^*$ 
using assms
proof (induct xvec rule: length-induct1)
  case Nil then show ?case by (auto intro: resCommRel-refl)
next
  case(Cons x xvec)
  note Cons1 = this
  show ?case
  proof (cases xvec)
    case Nil then show ?thesis
    by (auto intro: resCommRel-refl)
  next
  case(Cons y yvec)
  then have  $x \# \Psi$  and  $y \# \Psi$  and  $xvec \#* \Psi$  using  $\langle (x \# xvec) \#* \Psi \rangle$ 
  by simp+
  have  $(\Psi, (\nu * (x \# y \# yvec)) P, (\nu * (y \# x \# yvec)) P) \in \text{resCommRel}^*$  using  $\langle x \# \Psi \rangle$ 
   $\langle y \# \Psi \rangle$ 
  by (auto intro: resCommRel-swap)
  moreover have  $(\Psi, (\nu * (y \# x \# yvec)) P, (\nu * (y \# \text{rotate1 } (x \# yvec))) P) \in \text{resCommRel}^*$ 
  proof –
    have  $(\Psi, (\nu * (x \# yvec)) P, (\nu * \text{rotate1 } (x \# yvec)) P) \in \text{resCommRel}^*$  using
     $\langle xvec \#* \Psi \rangle$  Cons  $\langle x \# \Psi \rangle$ 
    by (intro Cons1) auto
    then show ?thesis using  $\langle y \# \Psi \rangle$ 
    unfolding resChain.simps
    by (rule resCommRelStarRes)
  qed
  ultimately show ?thesis unfolding Cons
  by (auto intro: rel-transcl-transitive)
qed

```

lemma *fresh-same-multiset*:

```

fixes  $xvec :: \text{name list}$ 
and  $yvec :: \text{name list}$ 
assumes  $\text{mset } xvec = \text{mset } yvec$ 
and  $xvec \#* X$ 

```

```

shows yvec #* X
proof -
  from ⟨mset xvec = mset yvec⟩ have set xvec = set yvec
    using mset-eq-setD by blast
  then show ?thesis using ⟨xvec #* X⟩
    unfolding fresh-def fresh-star-def name-list-supp
    by auto
qed

lemma nStepPerm-in-rel:
  assumes xvec #* Ψ
  shows (Ψ, (ν*xvec)P, (ν*(rotate n xvec))P) ∈ resCommRel*
  using assms
proof(induct n)
  case 0 then show ?case by(auto intro: resCommRel-refl)
next
  case(Suc n)
  then have (Ψ, (ν*xvec)P, (ν*rotate n xvec)P) ∈ resCommRel*
    by simp
  moreover have (Ψ, (ν*rotate n xvec)P, (ν*rotate (Suc n) xvec)P) ∈ resComm-
  Rel*
  proof -
    {
      fix xvec::name list
      assume xvec #* Ψ
      have rotate1 xvec #* Ψ using ⟨xvec #* Ψ⟩
      proof(induct xvec)
        case Nil then show ?case by simp
      next
        case Cons then show ?case by simp
      qed
    }
    note f1 = this
    have rotate n xvec #* Ψ using ⟨xvec #* Ψ⟩
      by(induct n) (auto simp add: f1)
    then show ?thesis
      by(auto simp add: oneStepPerm-in-rel)
  qed
  ultimately show ?case
    by(rule rel-trancl-transitive)
qed

lemma any-perm-in-rel:
  assumes xvec #* Ψ
  and mset xvec = mset yvec
  shows (Ψ, (ν*xvec)P, (ν*yvec)P) ∈ resCommRel*
  using assms
proof(induct xvec arbitrary: yvec rule: length-induct1)
  case Nil then show ?case by(auto intro: resCommRel.intros)

```

```

next
case(Cons x xvec)
obtain yvec1 yvec2 where yveceq: yvec=yvec1@x#yvec2
proof -
  assume 1:  $\bigwedge yvec1 yvec2. yvec = yvec1 @ x \# yvec2 \implies thesis$ 
  {
    note  $\langle mset (x \# xvec) = mset yvec \rangle$ 
    then have  $\exists yvec1 yvec2. yvec=yvec1@x\#yvec2$ 
    proof(induct xvec arbitrary: yvec rule: length-induct1)
      case Nil
      from  $\langle mset [x] = mset yvec \rangle$  have yvec = [x]
      apply(induct yvec)
      apply simp
      apply(simp add: single-is-union)
      done
    then show ?case by blast
  }
next
case(Cons x' xvec)
have less:  $\{\#x'\# \} \subseteq \# mset yvec$  unfolding  $\langle mset (x \# x' \# xvec) = mset$ 
yvec  $\rangle$ [symmetric]
  by simp
from  $\langle mset (x \# x' \# xvec) = mset yvec \rangle$ 
have  $mset (x \# xvec) = mset (remove1 x' yvec)$ 
  by (metis mset-remove1 remove1.simps(2))
then have  $\exists yvec1 yvec2. (remove1 x' yvec) = yvec1 @ x \# yvec2$ 
  by(intro Cons) simp+
then obtain yvec1 yvec2 where  $(remove1 x' yvec) = yvec1 @ x \# yvec2$ 
  by blast
then show ?case
proof(induct yvec arbitrary: yvec1 yvec2)
  case Nil then show ?case by simp
next
case(Cons y yvec) then show ?case
proof(cases x'=y)
  case True
  with  $\langle remove1 x' (y \# yvec) = yvec1 @ x \# yvec2 \rangle$ 
  have yvec = yvec1 @ x # yvec2
    by simp
  then show ?thesis
    by (metis append-Cons)
next
case False
then have  $remove1 x' (y\#yvec) = y \# remove1 x' (yvec)$ 
  by simp
note Cons2=Cons
show ?thesis
proof(cases yvec1)
  case Nil
  then show ?thesis using Cons2 False

```

```

      by auto
    next
      case(Cons y1 yvec1a)
        from ⟨remove1 x' (y # yvec) = yvec1 @ x # yvec2⟩ ⟨yvec1 = y1 #
yvec1a⟩ False have y1=y
          by simp
          from ⟨remove1 x' (y # yvec) = yvec1 @ x # yvec2⟩
          have remove1 x' yvec = yvec1a @ x # yvec2 unfolding Cons ⟨y1=y⟩
using False
            by simp
            then have ∃ yvec1 yvec2. yvec = yvec1 @ x # yvec2
            by(rule Cons2)
            then obtain yvec1 yvec2 where yvec = yvec1 @ x # yvec2
            by blast
            then show ?thesis
            by (metis append-Cons)
          qed
        qed
      qed
    qed
  }
  then show ?thesis using 1
  by blast
qed
from ⟨(x # xvec) #* Ψ⟩ have x # Ψ and xvec #* Ψ by auto
have mset (x # xvec) = mset(yvec1 @ x # yvec2)
  unfolding yveceq[symmetric] by fact
then have mset (xvec) = mset(yvec1 @ yvec2)
  by(subst add-right-cancel[symmetric,where a={#x#}]) (simp add: add.assoc)
then have (Ψ, (ν*xvec)P, (ν*(yvec1@yvec2))P) ∈ resCommRel* using ⟨xvec
#* Ψ⟩
  by(intro Cons) auto
moreover have (Ψ, (ν*(yvec1@yvec2))P, (ν*(yvec2@yvec1))P) ∈ resComm-
Rel*
proof –
  have e: yvec2 @ yvec1 = rotate (length yvec1) (yvec1@yvec2)
  apply(cases yvec2)
  apply simp
  apply(simp add: rotate-drop-take)
  done
  have (yvec1@yvec2) #* Ψ using ⟨mset (xvec) = mset(yvec1 @ yvec2)⟩ ⟨xvec
#* Ψ⟩
  by(rule fresh-same-multiset)
  then show ?thesis unfolding e
  by(rule nStepPerm-in-rel)
qed
ultimately have (Ψ, (ν*xvec)P, (ν*(yvec2 @ yvec1))P) ∈ resCommRel*
  by(rule rel-trancl-transitive)
then have (Ψ, (ν*(x#xvec))P, (ν*(x # yvec2 @ yvec1))P) ∈ resCommRel*

```

```

    unfolding resChain.simps using ⟨x # Ψ⟩
  by(rule resCommRelStarRes)
  moreover have (Ψ, (|ν*(x # yvec2 @ yvec1)|)P, (|ν*(yvec1 @ x # yvec2)|)P) ∈
resCommRel*
  proof -
    have e: yvec1 @ x # yvec2 = rotate (1+length yvec2) (x#yvec2@yvec1)
      apply(simp add: rotate-drop-take)
      apply(cases yvec1)
      by simp+
    have (yvec1 @ x # yvec2) #* Ψ using ⟨mset (x # xvec) = mset(yvec1 @ x #
yvec2)⟩ ⟨(x#xvec) #* Ψ⟩
      by(rule fresh-same-multiset)
    then have (x # yvec2 @ yvec1) #* Ψ by simp
    then show ?thesis unfolding e
      by(rule nStepPerm-in-rel)
  qed
  ultimately show ?case unfolding yveceq
    by(rule rel-trancl-transitive)
  qed

```

lemma bisimResComm:

```

  fixes x :: name
    and Ψ :: 'b
    and y :: name
    and P :: ('a, 'b, 'c) psi

```

shows $\Psi \triangleright (|\nu x)(|\nu y)P \sim (|\nu y)(|\nu x)P$

proof(cases x=y)

case True

then show ?thesis by(blast intro: bisimReflexive)

next

case False

{

fix x::name and y::name and P::('a, 'b, 'c) psi

assume x # Ψ and y # Ψ

let ?X = resCommRel

from ⟨x # Ψ⟩ ⟨y # Ψ⟩ have (Ψ, (|νx)(|\nu y)P, (|\nu y)(|\nu x)P) ∈ ?X

by(rule resCommRel-swap)

then have $\Psi \triangleright (|\nu x)(|\nu y)P \sim (|\nu y)(|\nu x)P$ using eqvtResCommRel

proof(coinduct rule: bisimStarWeakCoinduct)

case(cStatEq Ψ R S)

then show ?case

proof(induct rule: resCommRel.induct)

case resCommRel-refl then show ?case by(rule FrameStatEqRefl)

next

case(resCommRel-swap x Ψ y P)

moreover obtain $A_P \Psi_P$ where extractFrame P = ⟨ A_P, Ψ_P ⟩ and $A_P \#*$

Ψ and $x \# A_P$ and $y \# A_P$

by(rule freshFrame[where C=(x, y, Ψ)]) auto

```

ultimately show ?case by(force intro: frameResComm FrameStatEqTrans)
next
  case(resCommRel-res  $\Psi$   $P$   $Q$   $a$ )
  then show ?case by(auto intro: frameResPres)
qed
next
case(cSim  $\Psi$   $R$   $S$ )
have eqt(resCommRel*)
  by(rule rel-trancl-eqt) (rule eqtResCommRel)
from cSim show ?case
proof(induct rule: resCommRel.induct)
  case(resCommRel-refl  $\Psi$   $P$ )
  then show ?case
    by(rule simI[OF  $\langle$ eqt(resCommRel*) $\rangle$ ]) (blast intro: resCommRel.intros)
next
  case(resCommRel-swap  $a$   $\Psi$   $b$   $P$ )
  show ?case
    by(rule resComm) (fact|auto intro: resCommRel.intros any-perm-in-rel)+
next
  case(resCommRel-res  $\Psi$   $P$   $Q$   $a$ )
  show ?case
    by(rule resPres[where Rel=resCommRel*]) (fact|simp add: resCommRel-
StarResChain)+
qed
next
case(cExt  $\Psi$   $R$   $S$   $\Psi'$ ) then show ?case
proof(induct arbitrary:  $\Psi'$  rule: resCommRel.induct)
  case resCommRel-refl then show ?case by(rule resCommRel-refl)
next
  case(resCommRel-swap  $a$   $\Psi$   $b$   $P$ )
  then show ?case
  proof(cases  $a=b$ )
    case True show ?thesis unfolding  $\langle a=b \rangle$  by(rule resCommRel-refl)
  next
    case False
      obtain  $x::name$  and  $y::name$  where  $x \neq y$  and  $x \# \Psi$  and  $x \# \Psi'$  and  $x \#$ 
 $P$  and  $x \neq a$  and  $x \neq b$  and  $y \# \Psi$  and  $y \# \Psi'$  and  $y \# P$  and  $y \neq a$  and  $y \neq b$ 
        apply(generate-fresh name)
        apply(generate-fresh name)
        by force
      then show ?thesis using False
        apply(subst (1 2) alphaRes[where  $x=a$  and  $y=x$ ])
        apply assumption
        apply(simp add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst,
OF fin-suppl])
        apply(subst (1 2) alphaRes[where  $x=b$  and  $y=y$ ])
        apply(simp add: fresh-abs-fun-iff[OF pt-name-inst, OF at-name-inst,
OF fin-suppl] fresh-left)
        apply assumption

```



```

    unfolding eqvts
    apply(subst (1) cp1[OF cp-pt-inst, OF pt-name-inst, OF at-name-inst])
    by(auto simp add: eqvts swap-simps intro: resCommRel.intros)
qed
next
case(resCommRel-res  $\Psi$   $P$   $Q$   $a$ )
obtain  $b::name$  where  $b \# \Psi$  and  $b \neq a$  and  $b \# P$  and  $b \# Q$  and  $b \# \Psi'$ 
  by(generate-fresh name) auto
have  $(\Psi \otimes ((a,b) \cdot \Psi'), P, Q) \in resCommRel$  by fact
moreover from  $\langle b \# \Psi' \rangle$  have  $a \# ((a, b) \cdot \Psi')$  by(simp add: fresh-left
swap-simps)
with  $\langle a \# \Psi \rangle$  have  $a \# \Psi \otimes ((a,b) \cdot \Psi')$  by force
ultimately have  $(\Psi \otimes ((a,b) \cdot \Psi'), (\nu a)P, (\nu a)Q) \in resCommRel$ 
  by(rule resCommRel.intros)
then have  $((a,b) \cdot (\Psi \otimes ((a,b) \cdot \Psi'), (\nu a)P, (\nu a)Q)) \in resCommRel$  using
eqvResCommRel
  by(intro eqvI)
then have  $(\Psi \otimes \Psi', (\nu b)((a,b) \cdot P), (\nu b)((a,b) \cdot Q)) \in resCommRel$  using
 $\langle a \# \Psi \rangle \langle b \# \Psi \rangle$ 
  by(simp add: eqvts swap-simps)
then show ?case using  $\langle b \# Q \rangle \langle b \# P \rangle$ 
  apply(subst (1 2) alphaRes[where  $x=a$  and  $y=b$ ])
  by(assumption|simp only: eqvts)+
qed
next
case(cSym  $\Psi$   $R$   $S$ ) then show ?case
  by(induct rule: resCommRel.inducts) (auto intro: resCommRel.intros)
qed
}
moreover obtain  $x'::name$  where  $x' \# \Psi$  and  $x' \# P$  and  $x' \neq x$  and  $x' \neq y$ 
  by(generate-fresh name) auto
moreover obtain  $y'::name$  where  $y' \# \Psi$  and  $y' \# P$  and  $y' \neq x$  and  $y' \neq y$ 
and  $y' \neq x'$ 
  by(generate-fresh name) auto
ultimately have  $\Psi \triangleright ((\nu x')((\nu y')(((y, y'), (x, x')) \cdot P)) \sim ((\nu y')((\nu x')(((y, y'),
(x, x')) \cdot P)))$  by auto
then show ?thesis using  $\langle x' \# P \rangle \langle x' \neq x \rangle \langle x' \neq y \rangle \langle y' \# P \rangle \langle y' \neq x \rangle \langle y' \neq y \rangle
\langle y' \neq x' \rangle \langle x \neq y \rangle$ 
  apply(subst alphaRes[where  $x=x$  and  $y=x'$  and  $P=P$ ], auto)
  apply(subst alphaRes[where  $x=y$  and  $y=y'$  and  $P=P$ ], auto)
  apply(subst alphaRes[where  $x=x$  and  $y=x'$  and  $P=(\nu y')(((y, y')) \cdot P)$ ], auto
simp add: abs-fresh fresh-left)
  apply(subst alphaRes[where  $x=y$  and  $y=y'$  and  $P=(\nu x')(((x, x')) \cdot P)$ ], auto
simp add: abs-fresh fresh-left)
  by(subst perm-compose) (simp add: eqvts calc-atm)
qed

lemma bisimResComm':
  fixes  $x \quad :: name$ 

```

```

and  $\Psi$  :: 'b
and  $xvec$  :: name list
and  $P$  :: ('a, 'b, 'c) psi

assumes  $x \# \Psi$ 
and  $xvec \#* \Psi$ 

shows  $\Psi \triangleright (\nu x)((\nu *xvec)P) \sim (\nu *xvec)((\nu x)P)$ 
using assms
by(induct xvec) (auto intro: bisimResComm bisimReflexive bisimResPres bisim-
Transitive)

lemma bisimParPresSym:
fixes  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $Q$  :: ('a, 'b, 'c) psi
and  $R$  :: ('a, 'b, 'c) psi

assumes  $\Psi \triangleright P \sim Q$ 

shows  $\Psi \triangleright R \parallel P \sim R \parallel Q$ 
using assms
by(metis bisimParComm bisimParPres bisimTransitive)

lemma bisimScopeExt:
fixes  $x$  :: name
and  $\Psi$  :: 'b
and  $P$  :: ('a, 'b, 'c) psi
and  $Q$  :: ('a, 'b, 'c) psi

assumes  $x \# P$ 

shows  $\Psi \triangleright (\nu x)(P \parallel Q) \sim P \parallel (\nu x)Q$ 
proof -
{
fix  $x::name$  and  $Q$  :: ('a, 'b, 'c) psi
assume  $x \# \Psi$  and  $x \# P$ 
let  $?X1 = \{((\Psi::'b), (\nu *xvec)((\nu *yvec)((P::('a, 'b, 'c) psi) \parallel Q)), (\nu *xvec)(P$ 
 $\parallel ((\nu *yvec)Q)) \mid \Psi \ xvec \ yvec \ P \ Q. \ yvec \ \#* \ \Psi \ \wedge \ yvec \ \#* \ P \ \wedge \ xvec \ \#* \ \Psi\}$ 
let  $?X2 = \{((\Psi::'b), (\nu *xvec)((P::('a, 'b, 'c) psi) \parallel ((\nu *yvec)Q)), (\nu *xvec)((\nu *yvec)(P$ 
 $\parallel Q)) \mid \Psi \ xvec \ yvec \ P \ Q. \ yvec \ \#* \ \Psi \ \wedge \ yvec \ \#* \ P \ \wedge \ xvec \ \#* \ \Psi\}$ 
let  $?X = ?X1 \cup ?X2$ 
from  $\langle x \# \Psi \rangle \langle x \# P \rangle$  have  $(\Psi, (\nu x)(P \parallel Q), P \parallel (\nu x)Q) \in ?X$ 
proof -
from  $\langle x \# \Psi \rangle \langle x \# P \rangle$  have  $(\Psi, (\nu x)(P \parallel Q), P \parallel (\nu x)Q) \in ?X1$ 
by (smt (verit, del-insts) mem-Collect-eq freshSets(1) freshSets(5) resChain.base
resChain.step)
then show ?thesis by auto
qed

```

moreover have $eqvt \ ?X$
by (*rule eqvtUnion*) (*clarsimp simp add: eqvt-def eqvts, metis fresh-star-bij(2)*)+
ultimately have $\Psi \triangleright (\nu x)(P \parallel Q) \sim P \parallel (\nu x)Q$
proof(*coinduct rule: transitiveStarCoinduct*)
 case(*cStatEq* $\Psi \ R \ T$)
 then have $(\Psi, R, T) \in ?X1 \vee (\Psi, R, T) \in ?X2$
 by *blast*
 then show *?case*
 proof(*rule disjE*)
 assume $(\Psi, R, T) \in ?X1$
 then obtain $xvec \ yvec \ P \ Q$ **where** $R = (\nu *xvec)((\nu *yvec)(P \parallel Q))$ **and** $T = (\nu *xvec)(P \parallel ((\nu *yvec)Q))$ **and** $xvec \ \#* \ \Psi$ **and** $yvec \ \#* \ P$ **and** $yvec \ \#* \ \Psi$
 by *auto*
 moreover obtain $A_P \ \Psi_P$ **where** $FrP: extractFrame \ P = \langle A_P, \Psi_P \rangle$ **and** $A_P \ \#* \ \Psi$ **and** $yvec \ \#* \ A_P$ **and** $A_P \ \#* \ Q$
 by(*rule freshFrame[where C=($\Psi, yvec, Q$)]*) *auto*
 moreover obtain $A_Q \ \Psi_Q$ **where** $FrQ: extractFrame \ Q = \langle A_Q, \Psi_Q \rangle$ **and** $A_Q \ \#* \ \Psi$ **and** $yvec \ \#* \ A_Q$ **and** $A_Q \ \#* \ A_P$ **and** $A_Q \ \#* \ \Psi_P$
 by(*rule freshFrame[where C=($\Psi, yvec, A_P, \Psi_P$)]*) *auto*
 moreover from $FrQ \ \langle A_P \ \#* \ Q \rangle \ \langle A_Q \ \#* \ A_P \rangle$ **have** $A_P \ \#* \ \Psi_Q$
 by(*auto dest: extractFrameFreshChain*)
 moreover from $\langle yvec \ \#* \ P \rangle \ \langle yvec \ \#* \ A_P \rangle \ FrP$ **have** $yvec \ \#* \ \Psi_P$
 by(*drule-tac extractFrameFreshChain*) *auto*
 ultimately show *?case*
 by(*auto simp add: frameChainAppend[symmetric] freshChainAppend*) (*auto simp add: frameChainAppend intro: frameResChainPres frameResChainComm*)
 next
 assume $(\Psi, R, T) \in ?X2$
 then obtain $xvec \ yvec \ P \ Q$ **where** $T = (\nu *xvec)((\nu *yvec)(P \parallel Q))$ **and** $R = (\nu *xvec)(P \parallel ((\nu *yvec)Q))$ **and** $xvec \ \#* \ \Psi$ **and** $yvec \ \#* \ P$ **and** $yvec \ \#* \ \Psi$
 by *auto*
 moreover obtain $A_P \ \Psi_P$ **where** $FrP: extractFrame \ P = \langle A_P, \Psi_P \rangle$ **and** $A_P \ \#* \ \Psi$ **and** $yvec \ \#* \ A_P$ **and** $A_P \ \#* \ Q$
 by(*rule freshFrame[where C=($\Psi, yvec, Q$)]*) *auto*
 moreover obtain $A_Q \ \Psi_Q$ **where** $FrQ: extractFrame \ Q = \langle A_Q, \Psi_Q \rangle$ **and** $A_Q \ \#* \ \Psi$ **and** $yvec \ \#* \ A_Q$ **and** $A_Q \ \#* \ A_P$ **and** $A_Q \ \#* \ \Psi_P$
 by(*rule freshFrame[where C=($\Psi, yvec, A_P, \Psi_P$)]*) *auto*
 moreover from $FrQ \ \langle A_P \ \#* \ Q \rangle \ \langle A_Q \ \#* \ A_P \rangle$ **have** $A_P \ \#* \ \Psi_Q$
 by(*auto dest: extractFrameFreshChain*)
 moreover from $\langle yvec \ \#* \ P \rangle \ \langle yvec \ \#* \ A_P \rangle \ FrP$ **have** $yvec \ \#* \ \Psi_P$
 by(*drule-tac extractFrameFreshChain*) *auto*
 ultimately show *?case*
 by(*auto simp add: frameChainAppend[symmetric] freshChainAppend*) (*auto simp add: frameChainAppend intro: frameResChainPres frameResChainComm*)
 qed
 next
 case(*cSim* $\Psi \ R \ T$)
 let $?Y = \{(\Psi, P, Q) \mid \Psi \ P \ P' \ Q' \ Q. \ \Psi \triangleright \ P \sim P' \wedge ((\Psi, P', Q') \in ?X \vee \Psi \triangleright P' \sim Q') \wedge \Psi \triangleright Q' \sim Q\}$

from $\langle eqvt \ ?X \rangle$ **have** $eqvt \ ?Y$ **by** *blast*
have $C1: \bigwedge \Psi \ R \ T \ y. [(\Psi, R, T) \in ?Y; (y::name) \# \Psi] \implies (\Psi, (\nu y)R, (\nu y)T) \in ?Y$
proof –
fix $\Psi \ R \ T \ y$
assume $(\Psi, R, T) \in ?Y$
then obtain $R' \ T'$ **where** $\Psi \triangleright R \sim R'$ **and** $(\Psi, R', T') \in (?X \cup bisim)$
and $\Psi \triangleright T' \sim T$ **by** *force*
assume $(y::name) \# \Psi$
show $(\Psi, (\nu y)R, (\nu y)T) \in ?Y$
proof(*cases* $(\Psi, R', T') \in ?X$)
assume $(\Psi, R', T') \in ?X$
show *?thesis*
proof(*cases* $(\Psi, R', T') \in ?X1$)
assume $(\Psi, R', T') \in ?X1$
then obtain $xvec \ yvec \ P \ Q$ **where** $R'eq: R' = (\nu*xvec)((\nu*yvec)(P \parallel Q))$ **and** $T'eq: T' = (\nu*xvec)(P \parallel ((\nu*yvec)Q))$
and $xvec \#* \Psi$ **and** $yvec \#* P$ **and** $yvec \#* \Psi$
by *auto*
from $\langle \Psi \triangleright R \sim R' \rangle \langle y \# \Psi \rangle$ **have** $\Psi \triangleright (\nu y)R \sim (\nu y)R'$ **by**(*rule bisimResPres*)
moreover from $\langle xvec \#* \Psi \rangle \langle y \# \Psi \rangle \langle yvec \#* P \rangle \langle yvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*(y\#xvec))(\nu*yvec)(P \parallel Q), (\nu*(y\#xvec))(\nu*yvec)(P \parallel ((\nu*yvec)Q))) \in ?X1$
by(*force simp del: resChain.simps*)
with $R'eq \ T'eq$ **have** $(\Psi, (\nu y)R', (\nu y)T') \in ?X \cup bisim$ **by** *simp*
moreover from $\langle \Psi \triangleright T' \sim T \rangle \langle y \# \Psi \rangle$ **have** $\Psi \triangleright (\nu y)T' \sim (\nu y)T$
by(*rule bisimResPres*)
ultimately show *?thesis* **by** *blast*
next
assume $(\Psi, R', T') \notin ?X1$
with $\langle (\Psi, R', T') \in ?X \rangle$ **have** $(\Psi, R', T') \in ?X2$ **by** *blast*
then obtain $xvec \ yvec \ P \ Q$ **where** $T'eq: T' = (\nu*xvec)((\nu*yvec)(P \parallel Q))$ **and** $R'eq: R' = (\nu*xvec)(P \parallel ((\nu*yvec)Q))$ **and** $xvec \#* \Psi$ **and** $yvec \#* P$ **and** $yvec \#* \Psi$
by *auto*
from $\langle \Psi \triangleright R \sim R' \rangle \langle y \# \Psi \rangle$ **have** $\Psi \triangleright (\nu y)R \sim (\nu y)R'$ **by**(*rule bisimResPres*)
moreover from $\langle xvec \#* \Psi \rangle \langle y \# \Psi \rangle \langle yvec \#* P \rangle \langle yvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*(y\#xvec))(P \parallel ((\nu*yvec)Q)), (\nu*(y\#xvec))(\nu*yvec)(P \parallel Q)) \in ?X2$
by(*force simp del: resChain.simps*)
with $R'eq \ T'eq$ **have** $(\Psi, (\nu y)R', (\nu y)T') \in ?X \cup bisim$ **by** *simp*
moreover from $\langle \Psi \triangleright T' \sim T \rangle \langle y \# \Psi \rangle$ **have** $\Psi \triangleright (\nu y)T' \sim (\nu y)T$
by(*rule bisimResPres*)
ultimately show *?thesis* **by** *blast*
qed
next
assume $(\Psi, R', T') \notin ?X$
with $\langle (\Psi, R', T') \in ?X \cup bisim \rangle$ **have** $\Psi \triangleright R' \sim T'$ **by** *blast*
with $\langle \Psi \triangleright R \sim R' \rangle \langle \Psi \triangleright T' \sim T \rangle \langle y \# \Psi \rangle$ **show** *?thesis*

```

      by(blast dest: bisimResPres)
    qed
  qed
  have  $C1'$ :  $\bigwedge \Psi R T y. \llbracket (\Psi, R, T) \in ?Y^*; (y::name) \# \Psi \rrbracket \implies (\Psi, (\nu y)R, (\nu y)T) \in ?Y^*$ 
  proof -
    fix  $\Psi R$ 
    and  $T::('a,'b,'c) psi$ 
    and  $y::name$ 
    assume  $(\Psi, R, T) \in ?Y^*$ 
    and  $y \# \Psi$ 
    then show  $(\Psi, (\nu y)R, (\nu y)T) \in ?Y^*$ 
    proof(induct rule: rel-trancl.induct)
      case(r-into-rel-trancl  $\Psi P Q$ )
      then show ?case
        by (intro rel-trancl.intros(1)) (rule C1)
    next
      case(rel-trancl-into-rel-trancl  $\Psi P Q R$ )
      then show ?case
        using rel-trancl.intros(2) C1 by meson
    qed
  qed
  have  $C1''$ :  $\bigwedge \Psi R T yvec. \llbracket (\Psi, R, T) \in ?Y^*; (yvec::name list) \#* \Psi \rrbracket \implies (\Psi, (\nu*yvec)R, (\nu*yvec)T) \in ?Y^*$ 
  proof -
    fix  $\Psi R T$ 
    and  $yvec::name list$ 
    assume  $(\Psi, R, T) \in ?Y^*$ 
    and  $yvec \#* \Psi$ 
    then show  $(\Psi, (\nu*yvec)R, (\nu*yvec)T) \in ?Y^*$ 
    apply(induct yvec)
    apply simp
    unfolding resChain.simps
    by(rule C1') simp+
  qed
  have  $C2$ :  $\bigwedge y \Psi' R S zvec. \llbracket y \# \Psi'; y \# R; zvec \#* \Psi' \rrbracket \implies (\Psi', (\nu y)((\nu*zvec)(R \parallel S)), (\nu*zvec)(R \parallel (\nu y)S)) \in ?Y^*$ 
  proof -
    fix  $y::name$ 
    and  $\Psi::'b$ 
    and  $R::('a,'b,'c) psi$ 
    and  $S::('a,'b,'c) psi$ 
    and  $zvec::name list$ 
    assume  $y \# \Psi$ 
    and  $y \# R$ 
    and  $zvec \#* \Psi$ 
    have  $\Psi \triangleright (\nu y)((\nu*zvec)(R \parallel S)) \sim ((\nu*zvec)((\nu y)(R \parallel S)))$ 
    by(rule bisimResComm') fact+
    moreover have  $(\Psi, ((\nu*zvec)((\nu y)(R \parallel S))), (\nu*zvec)(R \parallel (\nu y)S)) \in ?X$ 

```

```

using ⟨y # Ψ⟩ ⟨y # R⟩ ⟨zvec #* Ψ⟩
  apply clarsimp
  apply(rule exI[where x=zvec])
  apply(rule exI[where x=[y]])
  by auto
  moreover have Ψ ▷ (ν*zvec)(R || (νy)S) ~ (ν*zvec)(R || (νy)S)
    by(rule bisimReflexive)
  ultimately show (Ψ, (νy)((ν*zvec)(R || S)), (ν*zvec)(R || (νy)S)) ∈ ?Y*
    by blast
qed
have C2': ∧y Ψ' R S zvec. [[y # Ψ'; y # R; zvec #* Ψ']] ⇒ (Ψ', (ν*zvec)(R ||
(νy)S), (νy)((ν*zvec)(R || S))) ∈ ?Y*
proof –
  fix y::name
  and Ψ::'b
  and R::('a,'b,'c) psi
  and S::('a,'b,'c) psi
  and zvec::name list
assume y # Ψ
  and y # R
  and zvec #* Ψ
have Ψ ▷ ((ν*zvec)((νy)(R || S))) ~ (νy)((ν*zvec)(R || S))
  by(rule bisimSymmetric[OF bisimResComm'] fact+)
moreover have (Ψ, (ν*zvec)(R || (νy)S), ((ν*zvec)((νy)(R || S)))) ∈ ?X
using ⟨y # Ψ⟩ ⟨y # R⟩ ⟨zvec #* Ψ⟩
  apply(intro UnI2)
  apply clarsimp
  apply(rule exI[where x=zvec])
  apply(rule exI[where x=[y]])
  by auto
  moreover have Ψ ▷ (ν*zvec)(R || (νy)S) ~ (ν*zvec)(R || (νy)S)
    by(rule bisimReflexive)
  ultimately show (Ψ, (ν*zvec)(R || (νy)S), (νy)((ν*zvec)(R || S))) ∈ ?Y*
    by blast
qed
have C3: ∧Ψ' zvec R y. [[y # Ψ'; zvec #* Ψ']] ⇒ (Ψ', (νy)((ν*zvec)R),
(ν*zvec)((νy)R)) ∈ ?Y*
proof –
  fix y::name
  and Ψ::'b
  and R::('a,'b,'c) psi
  and zvec::name list
assume y # Ψ
  and zvec #* Ψ
then have Ψ ▷ (νy)((ν*zvec)R) ~ (ν*zvec)((νy)R)
  by(rule bisimResComm')
then show (Ψ, (νy)((ν*zvec)R), (ν*zvec)((νy)R)) ∈ ?Y*
  by(blast intro: bisimReflexive)
qed

```

```

have C4:  $\bigwedge \Psi' R S \text{zvec}. \llbracket \text{zvec} \#* R; \text{zvec} \#* \Psi \rrbracket \implies (\Psi', ((\nu^* \text{zvec})(R \parallel S)),$ 
 $(R \parallel ((\nu^* \text{zvec})S))) \in ?Y^*$ 
proof –
  fix  $\Psi::'b$ 
    and  $R::('a, 'b, 'c) \text{psi}$ 
    and  $S::('a, 'b, 'c) \text{psi}$ 
    and  $\text{zvec}::\text{name list}$ 
  assume  $\text{zvec} \#* R$ 
    and  $\text{zvec} \#* \Psi$ 
  then have  $(\Psi, ((\nu^* \text{zvec})(R \parallel S)), (R \parallel ((\nu^* \text{zvec})S))) \in ?X$ 
    apply clarsimp
    apply(rule exI[where  $x=[]$ ])
    apply(rule exI[where  $x=\text{zvec}$ ])
    by auto
  then show  $(\Psi, ((\nu^* \text{zvec})(R \parallel S)), (R \parallel ((\nu^* \text{zvec})S))) \in ?Y^*$ 
    by(blast intro: bisimReflexive)
qed
have C4':  $\bigwedge \Psi' R S \text{zvec}. \llbracket \text{zvec} \#* R; \text{zvec} \#* \Psi \rrbracket \implies (\Psi', (R \parallel ((\nu^* \text{zvec})S)),$ 
 $((\nu^* \text{zvec})(R \parallel S))) \in ?Y^*$ 
proof –
  fix  $\Psi::'b$ 
    and  $R::('a, 'b, 'c) \text{psi}$ 
    and  $S::('a, 'b, 'c) \text{psi}$ 
    and  $\text{zvec}::\text{name list}$ 
  assume  $\text{zvec} \#* R$ 
    and  $\text{zvec} \#* \Psi$ 
  then have  $(\Psi, (R \parallel ((\nu^* \text{zvec})S)), ((\nu^* \text{zvec})(R \parallel S))) \in ?X$ 
    apply(intro UnI2)
    apply clarsimp
    apply(rule exI[where  $x=[]$ ])
    apply(rule exI[where  $x=\text{zvec}$ ])
    by auto
  then show  $(\Psi, (R \parallel ((\nu^* \text{zvec})S)), ((\nu^* \text{zvec})(R \parallel S))) \in ?Y^*$ 
    by(blast intro: bisimReflexive)
qed
{
  fix  $\Psi P Q R$ 
  assume  $\Psi \triangleright P \rightsquigarrow[?Y^*] Q$ 
    and  $\Psi \triangleright Q \rightsquigarrow[?Y^*] R$ 
  moreover note rel-trancl-eqvt[OF ‹eqvt ?Y›]
  moreover have  $\{(\Psi, P, R) \mid \Psi P R. \exists Q. (\Psi, P, Q) \in ?Y^* \wedge (\Psi, Q, R)$ 
 $\in ?Y^*\} \subseteq ?Y^*$ 
    by(auto intro: rel-trancl-transitive)
  ultimately have  $\Psi \triangleright P \rightsquigarrow[?Y^*] R$ 
    by(rule transitive)
}
note trans = this

show ?case

```

```

proof(cases (Ψ, R, T) ∈ ?X1)
  assume (Ψ, R, T) ∈ ?X1
  then obtain xvec yvec P Q where Req: R = (ν*xvec)((ν*yvec)(P || Q))
and Teq: T = (ν*xvec)(P || ((ν*yvec)Q)) and xvec #* Ψ and yvec #* P and yvec
#* Ψ
  by auto
  have Ψ ▷ (ν*xvec)((ν*yvec)(P || Q)) ~>[?Y*] (ν*xvec)(P || ((ν*yvec)Q))
  proof -
    have Ψ ▷ (ν*yvec)(P || Q) ~>[?Y*] P || ((ν*yvec)Q) using ⟨yvec #* Ψ⟩
    ⟨yvec #* P⟩
  proof(induct yvec arbitrary: Q)
    case Nil show ?case
      unfolding resChain.simps
      by(rule monotonic[where A=?Y]) (blast intro: reflexive bisimReflexive)+
    next
      case(Cons y yvec)
      then have yvec #* P and yvec #* Ψ and y # Ψ and y # P
      by simp+
      have Ψ ▷ (ν*(y#yvec))P || Q ~>[?Y*] (ν*yvec)((νy)(P || Q))
      proof -
        have Ψ ▷ (ν*(y#yvec))P || Q ~>[bisim] (ν*yvec)((νy)(P || Q))
          unfolding resChain.simps
          apply(rule bisimE)
          apply(rule bisimResComm')
          by fact+
        then have Ψ ▷ (ν*(y#yvec))P || Q ~>[?Y] (ν*yvec)((νy)(P || Q))
          apply -
          apply(drule monotonic[where B=?Y])
          by auto (blast intro: bisimTransitive bisimReflexive)
        then show ?thesis
          apply -
          apply(drule monotonic[where B=?Y*])
          by blast
      qed
      moreover have Ψ ▷ (ν*yvec)((νy)(P || Q)) ~>[?Y*] (ν*yvec)(P ||
(νy)Q)
      proof -
        have Ψ ▷ (νy)(P || Q) ~>[?Y*] P || (νy)Q
          apply(rule scopeExtLeft)
          apply fact
          apply fact
          apply(rule rel-trancl-eqt)
          apply fact
          apply(blast intro: bisimReflexive)
          by fact+
        then show ?thesis using ⟨yvec #* Ψ⟩
      proof(induct yvec)
        case Nil then show ?case by simp
      next

```



```

case(Cons y' yvec)
then show ?case
  unfolding resChain.simps
  apply -
  apply(rule resPres[where Rel=?Y* and Rel'=?Y*])
    apply simp
    apply(rule rel-trancl-eqvt[OF ‹eqvt ?Y›])
    apply simp
    apply simp
    by(rule C1'')
qed
qed
moreover have  $\Psi \triangleright (\nu * yvec)(P \parallel (\nu y) Q) \rightsquigarrow[?Y^*] P \parallel ((\nu * yvec)((\nu y) Q))$ 
  by(rule Cons) fact+
moreover have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[?Y^*] P \parallel ((\nu *(y\#yvec)) Q)$ 
proof -
  have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[bisim] P \parallel ((\nu *(y\#yvec)) Q)$ 
  unfolding resChain.simps
  apply(rule bisimE)
  apply(rule bisimParPresSym)
  apply(rule bisimSymmetric[OF bisimResComm])
  by fact+
  then have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[?Y] P \parallel ((\nu *(y\#yvec)) Q)$ 
  apply -
  apply(drule monotonic[where B=?Y])
  by auto (blast intro: bisimTransitive bisimReflexive)
  then show ?thesis
  apply -
  apply(drule monotonic[where B=?Y*])
  by blast
qed
moreover have  $\Psi \triangleright (\nu * yvec) P \parallel (\nu y) Q \rightsquigarrow[?Y^*] P \parallel ((\nu * yvec)((\nu y) Q))$ 
  by(rule Cons) fact+
moreover have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[?Y^*] P \parallel ((\nu *(y\#yvec)) Q)$ 
proof -
  have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[bisim] P \parallel ((\nu y)((\nu * yvec) Q))$ 
  apply(rule bisimE)
  apply(rule bisimParPresSym)
  apply(rule bisimSymmetric[OF bisimResComm])
  by fact+
  then have  $\Psi \triangleright P \parallel ((\nu * yvec)((\nu y) Q)) \rightsquigarrow[?Y] P \parallel ((\nu y)((\nu * yvec) Q))$ 
  apply -
  apply(drule monotonic[where B=?Y])
  by auto (blast intro: bisimTransitive bisimReflexive)
  then show ?thesis
  unfolding resChain.simps
  apply -
  apply(drule monotonic[where B=?Y*])
  by blast

```

```

qed
ultimately show ?case
  by(blast dest: trans)
qed
then show ?thesis using ⟨xvec #* Ψ⟩
  apply(induct xvec)
  apply simp
  unfolding resChain.simps
  apply(rule resPres)
  apply simp
  apply(rule rel-trancl-eqvt[OF ⟨eqvt ?Y⟩])
  apply simp
  apply simp
  apply(rule C1'')
  apply simp
  by assumption
qed
then show ?case unfolding Req Teq
  by -
next
assume (Ψ, R, T) ∉ ?X1
then have (Ψ, R, T) ∈ ?X2 using ⟨(Ψ, R, T) ∈ ?X⟩ by blast
then obtain xvec yvec P Q where Teq: T = (ν*xvec)((ν*yvec)(P || Q))
and Req: R = (ν*xvec)(P || ((ν*yvec)Q)) and xvec #* Ψ and yvec #* P and yvec
#* Ψ
  by auto
have Ψ ▷ (ν*xvec)(P || ((ν*yvec)Q)) ∼[?Y*] (ν*xvec)((ν*yvec)(P || Q))
proof -
  have Ψ ▷ P || ((ν*yvec)Q) ∼[?Y*] (ν*yvec)(P || Q) using ⟨yvec #* P⟩
  ⟨yvec #* Ψ⟩
proof(induct yvec arbitrary: Q)
  case Nil show ?case
    unfolding resChain.simps
  by(rule monotonic[where A=?Y]) (blast intro: reflexive bisimReflexive)+
next
  case(Cons y yvec)
  then have yvec #* P and yvec #* Ψ and y # Ψ and y # P
    by simp+
  have Ψ ▷ (ν*yvec)((νy)(P || Q)) ∼[?Y*] (ν*(y#yvec))P || Q
  proof -
    have Ψ ▷ (ν*yvec)((νy)(P || Q)) ∼[bisim] (ν*(y#yvec))P || Q
    unfolding resChain.simps
    apply(rule bisimE)
    apply(rule bisimSymmetric[OF bisimResComm'])
    by fact+
  then have Ψ ▷ (ν*yvec)((νy)(P || Q)) ∼[?Y] (ν*(y#yvec))P || Q
  apply -
  apply(drule monotonic[where B=?Y])
  by auto (blast intro: bisimTransitive bisimReflexive)

```

```

then show ?thesis
  apply -
  apply(drule monotonic[where B=?Y*])
  by blast
qed
moreover have  $\Psi \triangleright (\nu^*yvec)(P \parallel (\nu y)Q) \rightsquigarrow[?Y^*] (\nu^*yvec)((\nu y)(P \parallel$ 
Q))
proof -
have  $\Psi \triangleright P \parallel (\nu y)Q \rightsquigarrow[?Y^*] (\nu y)(P \parallel Q)$ 
  apply(rule scopeExtRight)
  apply fact
  apply fact
  apply(rule rel-trancl-eqt)
  apply fact
  apply(blast intro: bisimReflexive)
  by fact+
then show ?thesis using  $\langle yvec \#* \Psi \rangle$ 
proof(induct yvec)
  case Nil then show ?case by simp
next
  case(Cons y' yvec)
  then show ?case
    unfolding resChain.simps
    apply -
    apply(rule resPres[where Rel=?Y* and Rel'=?Y*])
    apply simp
    apply(rule rel-trancl-eqt[OF  $\langle eqvt ?Y \rangle$ ])
    apply simp
    apply simp
    by(rule C1'')
qed
moreover have  $\Psi \triangleright P \parallel ((\nu^*yvec)((\nu y)Q)) \rightsquigarrow[?Y^*] (\nu^*yvec)(P \parallel$ 
 $(\nu y)Q)$ 
  by(rule Cons) fact+
moreover have  $\Psi \triangleright P \parallel ((\nu^*(y\#yvec))Q) \rightsquigarrow[?Y^*] P \parallel ((\nu^*yvec)((\nu y)Q))$ 
proof -
have  $\Psi \triangleright P \parallel ((\nu^*(y\#yvec))Q) \rightsquigarrow[bisim] P \parallel ((\nu^*yvec)((\nu y)Q))$ 
  unfolding resChain.simps
  apply(rule bisimE)
  apply(rule bisimParPresSym)
  apply(rule bisimResComm')
  by fact+
then have  $\Psi \triangleright P \parallel ((\nu^*(y\#yvec))Q) \rightsquigarrow[?Y] P \parallel ((\nu^*yvec)((\nu y)Q))$ 
  apply -
  apply(drule monotonic[where B=?Y])
  by auto (blast intro: bisimTransitive bisimReflexive)
then show ?thesis
  apply -
  apply(drule monotonic[where B=?Y*])

```

```

      by blast
    qed
  qed
  moreover have  $\Psi \triangleright P \parallel ((\nu^*yvec)((\nu y) Q)) \rightsquigarrow[?Y^*] (\nu^*yvec)P \parallel (\nu y) Q$ 
    by(rule Cons) fact+
  moreover have  $\Psi \triangleright P \parallel ((\nu^*(y\#yvec)) Q) \rightsquigarrow[?Y^*] P \parallel ((\nu^*yvec)((\nu y) Q))$ 
  proof -
    have  $\Psi \triangleright P \parallel ((\nu y)((\nu^*yvec) Q)) \rightsquigarrow[bisim] P \parallel ((\nu^*yvec)((\nu y) Q))$ 
      apply(rule bisimE)
      apply(rule bisimParPresSym)
      apply(rule bisimResComm')
      by fact+
    then have  $\Psi \triangleright P \parallel ((\nu y)((\nu^*yvec) Q)) \rightsquigarrow[?Y] P \parallel ((\nu^*yvec)((\nu y) Q))$ 
      apply -
      apply(drule monotonic[where B=?Y])
      by auto (blast intro: bisimTransitive bisimReflexive)
    then show ?thesis
      unfolding resChain.simps
      apply -
      apply(drule monotonic[where B=?Y*])
      by blast
  qed
  ultimately show ?case
    by(blast dest: trans)
  qed
  then show ?thesis using  $\langle xvec \#* \Psi \rangle$ 
    apply(induct xvec)
    apply simp
    unfolding resChain.simps
    apply(rule resPres)
    apply simp
    apply(rule rel-trancl-eqvt[OF  $\langle eqvt ?Y \rangle$ ])
    apply simp
    apply simp
    apply(rule C1'')
    apply simp
    by assumption
  qed
  then show ?case unfolding Req Teq
    by -
  qed
next
case(cExt  $\Psi R T \Psi'$ )
show ?case
proof(cases  $(\Psi, R, T) \in ?X1$ )
  assume  $(\Psi, R, T) \in ?X1$ 
  then obtain  $xvec yvec P Q$  where Req:  $R = (\nu^*xvec)((\nu^*yvec)(P \parallel Q))$ 
and Teq:  $T = (\nu^*xvec)(P \parallel ((\nu^*yvec) Q))$  and  $xvec \#* \Psi$  and  $yvec \#* P$  and  $yvec \#* \Psi'$ 

```

```

    by auto
  obtain p where (p · yvec) #* Ψ and (p · yvec) #* P and (p · yvec) #* Q
and (p · yvec) #* Ψ' and (p · yvec) #* xvec
  and S: (set p) ⊆ (set yvec) × (set (p · yvec)) and distinctPerm p
  by(rule name-list-avoiding[where c=(Ψ, P, Q, xvec, Ψ')]) auto
  obtain q where (q · xvec) #* Ψ and (q · xvec) #* Ψ' and (q · xvec) #* P
and (q · xvec) #* yvec and (q · xvec) #* Q and (q · xvec) #* (p·P) and (q · xvec)
#* (p·Q) and distinctPerm q and T: (set q) ⊆ (set xvec) × (set(q·xvec)) and (q
· xvec) #* (p·yvec)
  by(rule name-list-avoiding[where c=(Ψ, P, Q, yvec, p·yvec, Ψ',p·P,p·Q)])
auto
  note ⟨(p · yvec) #* Q⟩ ⟨set p ⊆ set yvec × set (p · yvec)⟩
  moreover have (p · yvec) #* (P || Q) using ⟨(p · yvec) #* Q⟩ ⟨(p · yvec)
#* P⟩
  by simp
  moreover have (Ψ ⊗ Ψ', (ν*xvec)(ν*p · yvec)p · P || Q, (ν*xvec)P ||
((ν*p · yvec)p · Q)) ∈ ?X
  proof –
    have Pdef: (p · P) = P using ⟨set p ⊆ set yvec × set(p·yvec)⟩ ⟨yvec #*
P⟩ ⟨(p·yvec) #* P⟩
    by simp
    have yvecdef: (q · p · yvec) = (p · yvec) using ⟨set q ⊆ set xvec ×
set(q·xvec)⟩ ⟨(p·yvec) #* xvec⟩ ⟨(q·xvec) #* (p·yvec)⟩
    by simp
    have (q · xvec) #* (P || ((ν*p · yvec)p · Q)) using ⟨(q · xvec) #* P⟩ ⟨(q ·
xvec) #* (p·Q)⟩ ⟨(q · xvec) #* (p·yvec)⟩
    by simp
    moreover have (q · xvec) #* ((ν*p · yvec)p · P || Q)
    unfolding eqvts Pdef
    using ⟨(q · xvec) #* P⟩ ⟨(q · xvec) #* (p·Q)⟩ ⟨(q · xvec) #* (p·yvec)⟩
    by simp
    moreover note ⟨set q ⊆ set xvec × set (q · xvec)⟩
    moreover have (Ψ ⊗ Ψ', (ν*q · xvec)q · (ν*p · yvec)p · P || Q,
(ν*q · xvec)q · P || ((ν*p · yvec)p · Q)) ∈ ?X
  proof –
    have (p·yvec) #* (Ψ⊗Ψ') using ⟨(p·yvec) #* Ψ⟩ ⟨(p·yvec) #* Ψ'⟩
    by auto
    moreover have (p·yvec) #* (q·P) using ⟨(p·yvec) #* P⟩
    apply(subst yvecdef[symmetric])
    by(subst fresh-star-bij)
    moreover have (q·xvec) #* (Ψ⊗Ψ') using ⟨(q·xvec) #* Ψ⟩ ⟨(q·xvec) #*
Ψ'⟩
    by auto
  ultimately show ?thesis
  unfolding Pdef eqvts yvecdef
  by blast
qed
ultimately show ?thesis
by(subst (1 2) resChainAlpha[where p=q and xvec=xvec])

```

```

qed
ultimately show ?case unfolding Req Teq
  apply(intro disjI1)
  by(subst (1 2) resChainAlpha[where p=p and yvec=yvec])
next
assume (Ψ, R, T) ∉ ?X1
then have (Ψ, R, T) ∈ ?X2 using ⟨(Ψ,R,T) ∈ ?X⟩
  by blast
then obtain xvec yvec P Q where Teq: T = (ν*xvec)((ν*yvec)(P || Q))
and Req: R = (ν*xvec)(P || ((ν*yvec)Q)) and xvec ‡* Ψ and yvec ‡* P and yvec
‡* Ψ
  by auto
  obtain p where (p · yvec) ‡* Ψ and (p · yvec) ‡* P and (p · yvec) ‡* Q
and (p · yvec) ‡* Ψ' and (p · yvec) ‡* xvec
  and S: (set p) ⊆ (set yvec) × (set(p · yvec)) and distinctPerm p
  by(rule name-list-avoiding[where c=(Ψ, P, Q, xvec, Ψ')]) auto
  obtain q where (q · xvec) ‡* Ψ and (q · xvec) ‡* Ψ' and (q · xvec) ‡* P
and (q · xvec) ‡* yvec and (q · xvec) ‡* Q and (q · xvec) ‡* (p·P) and (q · xvec)
‡* (p·Q) and distinctPerm q and T: (set q) ⊆ (set xvec) × (set(q·xvec)) and (q
· xvec) ‡* (p·yvec)
  by(rule name-list-avoiding[where c=(Ψ, P, Q, yvec, p·yvec, Ψ',p·P,p·Q)])
auto
note ⟨(p · yvec) ‡* Q⟩ ⟨set p ⊆ set yvec × set (p · yvec)⟩
moreover have (p · yvec) ‡* (P || Q) using ⟨(p · yvec) ‡* Q⟩ ⟨(p · yvec)
‡* P⟩
  by simp
moreover have (Ψ ⊗ Ψ', (ν*xvec)P || ((ν*p · yvec)p · Q), (ν*xvec)(ν*p
· yvec)p · P || Q) ∈ ?X
proof -
  have Pdef: (p · P) = P using ⟨set p ⊆ set yvec × set(p·yvec)⟩ ⟨yvec ‡*
P⟩ ⟨(p·yvec) ‡* P⟩
  by simp
  have yvecdef: (q · p · yvec) = (p · yvec) using ⟨set q ⊆ set xvec ×
set(q·xvec)⟩ ⟨(p·yvec) ‡* xvec⟩ ⟨(q·xvec) ‡* (p·yvec)⟩
  by simp
  have (q · xvec) ‡* (P || ((ν*p · yvec)p · Q)) using ⟨(q · xvec) ‡* P⟩ ⟨(q ·
xvec) ‡* (p·Q)⟩ ⟨(q · xvec) ‡* (p·yvec)⟩
  by simp
  moreover have (q · xvec) ‡* ((ν*p · yvec)p · P || Q)
  unfolding eqvts Pdef
  using ⟨(q · xvec) ‡* P⟩ ⟨(q · xvec) ‡* (p·Q)⟩ ⟨(q · xvec) ‡* (p·yvec)⟩
  by simp
  moreover note ⟨set q ⊆ set xvec × set (q · xvec)⟩
  moreover have (Ψ ⊗ Ψ', (ν*q · xvec)q · P || ((ν*p · yvec)p · Q), (ν*q
· xvec)q · (ν*p · yvec)p · P || Q) ∈ ?X
proof -
  have (p·yvec) ‡* (Ψ⊗Ψ') using ⟨(p·yvec) ‡* Ψ⟩ ⟨(p·yvec) ‡* Ψ'⟩
  by auto
  moreover have (p·yvec) ‡* (q·P) using ⟨(p·yvec) ‡* P⟩

```

```

      apply(subst yvecdef[symmetric])
      by(subst fresh-star-bij)
    moreover have (q·xvec) #* (Ψ⊗Ψ') using ⟨(q·xvec) #* Ψ⟩ ⟨(q·xvec) #*
Ψ'⟩
      by auto
      ultimately show ?thesis
      unfolding Pdef eqvts yvecdef
      by blast
    qed
    ultimately show ?thesis
    by(subst (1 2) resChainAlpha[where p=q and xvec=xvec])
  qed
  ultimately show ?case unfolding Req Teq
  apply(intro disjI1)
  by(subst (1 2) resChainAlpha[where p=p and xvec=yvec])
  qed
next
case(cSym Ψ P Q)
then show ?case
  by(blast dest: bisimE)
  qed
}
moreover obtain y::name where y # Ψ and y # P y # Q
  by(generate-fresh name) auto
ultimately have Ψ ▷ (νy)(P || [(x, y)] · Q) ~ P || (νy)([(x, y)] · Q) by auto
then show ?thesis using assms ⟨y # P⟩ ⟨y # Q⟩
  apply(subst alphaRes[where x=x and y=y and P=Q], auto)
  by(subst alphaRes[where x=x and y=y and P=P || Q]) auto
qed

```

lemma *bisimScopeExtChain:*

```

  fixes xvec :: name list
  and Ψ      :: 'b
  and P :: ('a, 'b, 'c) psi
  and Q :: ('a, 'b, 'c) psi

```

```

assumes xvec #* Ψ
  and   xvec #* P

```

shows $\Psi \triangleright (\nu * xvec)(P \parallel Q) \sim P \parallel ((\nu * xvec) Q)$

```

  using assms
  by(induct xvec) (auto intro: bisimScopeExt bisimReflexive bisimTransitive bisim-
ResPres)

```

lemma *bisimParAssoc:*

```

  fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi
  and Q :: ('a, 'b, 'c) psi
  and R :: ('a, 'b, 'c) psi

```

shows $\Psi \triangleright (P \parallel Q) \parallel R \sim P \parallel (Q \parallel R)$
proof –
let $?X = \{(\Psi, (\nu^*xvec)((P \parallel Q) \parallel R), (\nu^*xvec)(P \parallel (Q \parallel R))) \mid \Psi \text{ xvec } P \text{ } Q \text{ } R. \text{ xvec } \#^* \Psi\}$
let $?Y = \{(\Psi, P, Q) \mid \Psi \text{ } P \text{ } P' \text{ } Q' \text{ } Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in ?X \wedge \Psi \triangleright Q' \sim Q\}$

have $(\Psi, (P \parallel Q) \parallel R, P \parallel (Q \parallel R)) \in ?X$
by (*clarsimp*, *rule exI*[**where** $x = []$]) *auto*
moreover have *eqvt* $?X$ **by** (*force simp add: eqvt-def simp add: pt-fresh-star-bij*[*OF pt-name-inst, OF at-name-inst*] *eqvts*)
ultimately show *?thesis*
proof (*coinduct rule: weakTransitiveCoinduct*[^])
case (*cStatEq* $\Psi \text{ } PQR \text{ } PQR'$)
from $\langle (\Psi, PQR, PQR') \in ?X \rangle$ **obtain** *xvec* $P \text{ } Q \text{ } R$ **where** *xvec* $\#^* \Psi$ **and** $PQR = (\nu^*xvec)((P \parallel Q) \parallel R)$ **and** $PQR' = (\nu^*xvec)(P \parallel (Q \parallel R))$
by *auto*
moreover obtain $A_P \text{ } \Psi_P$ **where** *FrP*: *extractFrame* $P = \langle A_P, \Psi_P \rangle$ **and** $A_P \#^* \Psi$ **and** $A_P \#^* Q$ **and** $A_P \#^* R$
by (*rule freshFrame*[**where** $C = (\Psi, Q, R)$]) *auto*
moreover obtain $A_Q \text{ } \Psi_Q$ **where** *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$ **and** $A_Q \#^* \Psi$ **and** $A_Q \#^* A_P$ **and** $A_Q \#^* \Psi_P$ **and** $A_Q \#^* R$
by (*rule freshFrame*[**where** $C = (\Psi, A_P, \Psi_P, R)$]) *auto*
moreover obtain $A_R \text{ } \Psi_R$ **where** *FrR*: *extractFrame* $R = \langle A_R, \Psi_R \rangle$ **and** $A_R \#^* \Psi$ **and** $A_R \#^* A_P$ **and** $A_R \#^* \Psi_P$ **and** $A_R \#^* A_Q$ **and** $A_R \#^* \Psi_Q$
by (*rule freshFrame*[**where** $C = (\Psi, A_P, \Psi_P, A_Q, \Psi_Q)$]) *auto*
moreover from *FrQ* $\langle A_P \#^* Q \rangle \langle A_Q \#^* A_P \rangle$ **have** $A_P \#^* \Psi_Q$
by (*auto dest: extractFrameFreshChain*)
moreover from *FrR* $\langle A_P \#^* R \rangle \langle A_R \#^* A_P \rangle$ **have** $A_P \#^* \Psi_R$
by (*auto dest: extractFrameFreshChain*)
moreover from *FrR* $\langle A_Q \#^* R \rangle \langle A_R \#^* A_Q \rangle$ **have** $A_Q \#^* \Psi_R$
by (*auto dest: extractFrameFreshChain*)
ultimately show *?case using freshCompChain*
by *auto* (*metis frameChainAppend compositionSym Associativity frameNil-StatEq frameResChainPres*)

next
case (*cSim* $\Psi \text{ } T \text{ } S$)
from $\langle (\Psi, T, S) \in ?X \rangle$ **obtain** *xvec* $P \text{ } Q \text{ } R$ **where** *xvec* $\#^* \Psi$ **and** *TEq*: $T = (\nu^*xvec)((P \parallel Q) \parallel R)$
and *SEq*: $S = (\nu^*xvec)(P \parallel (Q \parallel R))$
by *auto*
from $\langle \text{eqvt } ?X \rangle$ **have** *eqvt* $?Y$ **by** *blast*
have $C1: \bigwedge \Psi \text{ } T \text{ } S \text{ yvec. } \llbracket (\Psi, T, S) \in ?Y; \text{ yvec } \#^* \Psi \rrbracket \implies (\Psi, (\nu^*yvec)T, (\nu^*yvec)S) \in ?Y$
proof –
fix $\Psi \text{ } T \text{ } S \text{ yvec}$
assume $(\Psi, T, S) \in ?Y$
then obtain $T' \text{ } S'$ **where** $\Psi \triangleright T \sim T'$ **and** $(\Psi, T', S') \in ?X$ **and** $\Psi \triangleright S'$

$\sim S$ **by force**
assume $\langle yvec :: name\ list \rangle \#* \Psi$
from $\langle (\Psi, T', S') \in ?X \rangle$ **obtain** $xvec\ P\ Q\ R$ **where** $T'eq: T' = (\nu*xvec)((P \parallel Q) \parallel R)$ **and** $S'eq: S' = (\nu*xvec)(P \parallel (Q \parallel R))$
and $xvec \#* \Psi$
by auto
from $\langle \Psi \triangleright T \sim T' \rangle$ $\langle yvec \#* \Psi \rangle$ **have** $\Psi \triangleright (\nu*yvec)T \sim (\nu*yvec)T'$ **by** $(rule\ bisimResChainPres)$
moreover from $\langle xvec \#* \Psi \rangle$ $\langle yvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*(yvec@xvec))((P \parallel Q) \parallel R), (\nu*(yvec@xvec))(P \parallel (Q \parallel R))) \in ?X$
by force
with $T'eq\ S'eq$ **have** $(\Psi, (\nu*yvec)T', (\nu*yvec)S') \in ?X$ **by** $(simp\ add: resChainAppend)$
moreover from $\langle \Psi \triangleright S' \sim S \rangle$ $\langle yvec \#* \Psi \rangle$ **have** $\Psi \triangleright (\nu*yvec)S' \sim (\nu*yvec)S$ **by** $(rule\ bisimResChainPres)$
ultimately show $(\Psi, (\nu*yvec)T, (\nu*yvec)S) \in ?Y$ **by blast**
qed

{
fix y
have $\bigwedge \Psi\ T\ S. \llbracket (\Psi, T, S) \in ?Y; y \#* \Psi \rrbracket \implies (\Psi, (\nu y)T, (\nu y)S) \in ?Y$
by $(drule\ C1[where\ yvec2=[y]])\ auto$
}
note $C2 = this$

have $\Psi \triangleright (\nu*xvec)((P \parallel Q) \parallel R) \rightsquigarrow[?Y] (\nu*xvec)(P \parallel (Q \parallel R))$
proof –
have $\Psi \triangleright (P \parallel Q) \parallel R \rightsquigarrow[?Y] P \parallel (Q \parallel R)$
proof –
note $\langle eqvt\ ?Y \rangle$
moreover have $\bigwedge \Psi\ P\ Q\ R. (\Psi, (P \parallel Q) \parallel R, P \parallel (Q \parallel R)) \in ?Y$
proof –
fix $\Psi\ P\ Q\ R$
have $(\Psi :: 'b, ((P :: ('a, 'b, 'c)\ psi) \parallel Q) \parallel R, P \parallel (Q \parallel R)) \in ?X$
by $(clarsimp, rule\ exI[where\ x=[]])\ auto$
then show $(\Psi, (P \parallel Q) \parallel R, P \parallel (Q \parallel R)) \in ?Y$
by $(blast\ intro: bisimReflexive)$
qed
moreover have $\bigwedge xvec\ \Psi\ P\ Q\ R. \llbracket xvec \#* \Psi; xvec \#* P \rrbracket \implies (\Psi, (\nu*xvec)((P \parallel Q) \parallel R), P \parallel ((\nu*xvec)(Q \parallel R))) \in ?Y$
proof –
fix $xvec\ \Psi\ P\ Q\ R$
assume $\langle xvec :: name\ list \rangle \#* (\Psi :: 'b)$ **and** $xvec \#* (P :: ('a, 'b, 'c)\ psi)$
from $\langle xvec \#* \Psi \rangle$ **have** $(\Psi, (\nu*xvec)((P \parallel Q) \parallel R), (\nu*xvec)(P \parallel (Q \parallel R))) \in ?X$ **by blast**
moreover from $\langle xvec \#* \Psi \rangle$ $\langle xvec \#* P \rangle$ **have** $\Psi \triangleright (\nu*xvec)(P \parallel (Q \parallel R)) \sim P \parallel ((\nu*xvec)(Q \parallel R))$
by $(rule\ bisimScopeExtChain)$
ultimately show $(\Psi, (\nu*xvec)((P \parallel Q) \parallel R), P \parallel ((\nu*xvec)(Q \parallel R))) \in$

```

?Y
  by(blast intro: bisimReflexive)
qed
  moreover have  $\bigwedge xvec \Psi P Q R. [\![xvec \#* \Psi; xvec \#* R]\!] \implies (\Psi, (\nu*xvec)(P \parallel Q)) \parallel R, (\nu*xvec)(P \parallel (Q \parallel R)) \in ?Y$ 
  proof -
    fix xvec  $\Psi P Q R$ 
    assume (xvec::name list)  $\#* (\Psi::'b)$  and xvec  $\#* (R::('a, 'b, 'c) psi)$ 
    have  $\Psi \triangleright ((\nu*xvec)(P \parallel Q)) \parallel R \sim R \parallel ((\nu*xvec)(P \parallel Q))$  by(rule bisimParComm)
    moreover from  $\langle xvec \#* \Psi \rangle \langle xvec \#* R \rangle$  have  $\Psi \triangleright (\nu*xvec)(R \parallel (P \parallel Q)) \sim R \parallel ((\nu*xvec)(P \parallel Q))$  by(rule bisimScopeExtChain)
    then have  $\Psi \triangleright R \parallel ((\nu*xvec)(P \parallel Q)) \sim (\nu*xvec)(R \parallel (P \parallel Q))$  by(rule bisimE)
    moreover from  $\langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu*xvec)(R \parallel (P \parallel Q)) \sim (\nu*xvec)((P \parallel Q) \parallel R)$ 
    by(metis bisimResChainPres bisimParComm)
    moreover from  $\langle xvec \#* \Psi \rangle$  have  $(\Psi, (\nu*xvec)((P \parallel Q) \parallel R), (\nu*xvec)(P \parallel (Q \parallel R))) \in ?X$  by blast
    ultimately show  $(\Psi, ((\nu*xvec)(P \parallel Q)) \parallel R, (\nu*xvec)(P \parallel (Q \parallel R))) \in ?Y$  by(blast dest: bisimTransitive intro: bisimReflexive)
  qed
  ultimately show ?thesis using C1
  by(rule parAssocLeft)
qed
then show ?thesis using  $\langle eqvt ?Y \rangle \langle xvec \#* \Psi \rangle C1$ 
by(rule resChainPres)
qed
with TEq SEq show ?case by simp
next
case(cExt  $\Psi T S \Psi'$ )
from  $\langle (\Psi, T, S) \in ?X \rangle$  obtain xvec P Q R where xvec  $\#* \Psi$  and TEq:  $T = (\nu*xvec)((P \parallel Q) \parallel R)$ 
and SEq:  $S = (\nu*xvec)(P \parallel (Q \parallel R))$ 
by auto
obtain p where  $(p \cdot xvec) \#* \Psi$  and  $(p \cdot xvec) \#* P$  and  $(p \cdot xvec) \#* Q$  and  $(p \cdot xvec) \#* R$  and  $(p \cdot xvec) \#* \Psi'$ 
and  $S: (set p) \subseteq (set xvec) \times (set(p \cdot xvec))$  and distinctPerm p
by(rule name-list-avoiding[where c=( $\Psi, P, Q, R, \Psi'$ )]) auto

from  $\langle (p \cdot xvec) \#* \Psi \rangle \langle (p \cdot xvec) \#* \Psi' \rangle$  have  $(\Psi \otimes \Psi', (\nu*(p \cdot xvec))(((p \cdot P) \parallel (p \cdot Q)) \parallel (p \cdot R)), (\nu*(p \cdot xvec))(((p \cdot P) \parallel ((p \cdot Q) \parallel (p \cdot R)))) \in ?X$ 
by auto
moreover from TEq  $\langle (p \cdot xvec) \#* P \rangle \langle (p \cdot xvec) \#* Q \rangle \langle (p \cdot xvec) \#* R \rangle S$ 
have  $T = (\nu*(p \cdot xvec))(((p \cdot P) \parallel (p \cdot Q)) \parallel (p \cdot R))$ 
apply clarsimp by(subst resChainAlpha[of p]) auto
moreover from SEq  $\langle (p \cdot xvec) \#* P \rangle \langle (p \cdot xvec) \#* Q \rangle \langle (p \cdot xvec) \#* R \rangle S$ 
have  $S = (\nu*(p \cdot xvec))(((p \cdot P) \parallel ((p \cdot Q) \parallel (p \cdot R))))$ 
apply clarsimp by(subst resChainAlpha[of p]) auto

```

```

ultimately show ?case by simp
next
case(cSym  $\Psi$   $T$   $S$ )
from  $\langle (\Psi, T, S) \in ?X \rangle$  obtain  $xvec$   $P$   $Q$   $R$  where  $xvec \#* \Psi$  and  $TEq: T =$ 
 $(\nu*xvec)((P \parallel Q) \parallel R)$ 
and  $SEq: (\nu*xvec)(P \parallel (Q \parallel R)) = S$ 
by auto

from  $\langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu*xvec)(P \parallel (Q \parallel R)) \sim (\nu*xvec)((R \parallel Q) \parallel P)$ 
by(metis bisimParComm bisimParPres bisimTransitive bisimResChainPres)
moreover from  $\langle xvec \#* \Psi \rangle$  have  $(\Psi, (\nu*xvec)((R \parallel Q) \parallel P), (\nu*xvec)(R \parallel$ 
 $(Q \parallel P))) \in ?X$  by blast
moreover from  $\langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu*xvec)(R \parallel (Q \parallel P)) \sim (\nu*xvec)((P$ 
 $\parallel Q) \parallel R)$ 
by(metis bisimParComm bisimParPres bisimTransitive bisimResChainPres)
ultimately show ?case using  $TEq$   $SEq$  by(blast dest: bisimTransitive)
qed
qed

```

lemma *bisimParNil*:

fixes $P :: ('a, 'b, 'c)$ psi

shows $\Psi \triangleright P \parallel \mathbf{0} \sim P$

proof –

let $?X1 = \{(\Psi, P \parallel \mathbf{0}, P) \mid \Psi P. True\}$

let $?X2 = \{(\Psi, P, P \parallel \mathbf{0}) \mid \Psi P. True\}$

let $?X = ?X1 \cup ?X2$

have *eqvt* $?X$ by(auto simp add: eqvt-def)

have $(\Psi, P \parallel \mathbf{0}, P) \in ?X$ by simp

then show ?thesis

proof(coinduct rule: bisimWeakCoinduct)

case(cStatEq Ψ Q R)

show ?case

proof(cases $(\Psi, Q, R) \in ?X1$)

assume $(\Psi, Q, R) \in ?X1$

then obtain P where $Q = P \parallel \mathbf{0}$ and $R = P$ by auto

moreover obtain A_P Ψ_P where *extractFrame* $P = \langle A_P, \Psi_P \rangle$ and $A_P \#* \Psi$
by(rule freshFrame)

ultimately show ?case

apply *clarsimp* by(metis frameResChainPres frameNilStatEq Identity Associativity Assertion.StatEqTrans Commutativity)

next

assume $(\Psi, Q, R) \notin ?X1$

with $\langle (\Psi, Q, R) \in ?X \rangle$ have $(\Psi, Q, R) \in ?X2$ by blast

then obtain P where $Q = P$ and $R = P \parallel \mathbf{0}$ by auto

moreover obtain A_P Ψ_P where *extractFrame* $P = \langle A_P, \Psi_P \rangle$ and $A_P \#* \Psi$
by(rule freshFrame)

ultimately show ?case

apply *clarsimp* by(metis frameResChainPres frameNilStatEq Identity Asso-

```

ciativity AssertionStatEqTrans AssertionStatEqSym Commutativity)
  qed
next
  case(cSim  $\Psi$   $Q$   $R$ )
  then show ?case using ‹eqvt ? $X$ ›
    by(auto intro: parNilLeft parNilRight)
next
  case(cExt  $\Psi$   $Q$   $R$   $\Psi'$ )
  then show ?case by auto
next
  case(cSym  $\Psi$   $Q$   $R$ )
  then show ?case by auto
qed
qed

lemma bisimResNil:
  fixes  $x :: \text{name}$ 
  and  $\Psi :: 'b$ 

shows  $\Psi \triangleright (\nu x)\mathbf{0} \sim \mathbf{0}$ 
proof -
  {
    fix  $x :: \text{name}$ 
    assume  $x \# \Psi$ 
    have  $\Psi \triangleright (\nu x)\mathbf{0} \sim \mathbf{0}$ 
    proof -
      let ? $X1$  = { $(\Psi, (\nu x)\mathbf{0}, \mathbf{0}) \mid \Psi \ x. \ x \# \Psi$ }
      let ? $X2$  = { $(\Psi, \mathbf{0}, (\nu x)\mathbf{0}) \mid \Psi \ x. \ x \# \Psi$ }
      let ? $X$  = ? $X1 \cup ?X2$ 

      from ‹ $x \# \Psi$ › have  $(\Psi, (\nu x)\mathbf{0}, \mathbf{0}) \in ?X$  by auto
      then show ?thesis
      proof(coinduct rule: bisimWeakCoinduct)
        case(cStatEq  $\Psi$   $P$   $Q$ )
        then show ?case using freshComp by(force intro: frameResFresh FrameS-
tatEqSym)
      next
        case(cSim  $\Psi$   $P$   $Q$ )
        then show ?case
          by(force intro: resNilLeft resNilRight)
      next
        case(cExt  $\Psi$   $P$   $Q$   $\Psi'$ )
        obtain  $y$  where  $y \# \Psi$  and  $y \# \Psi'$  and  $y \neq x$ 
          by(generate-fresh name) (auto simp add: fresh-prod)
        show ?case
        proof(cases  $(\Psi, P, Q) \in ?X1$ )
          assume  $(\Psi, P, Q) \in ?X1$ 
          then obtain  $x$  where  $P = (\nu x)\mathbf{0}$  and  $Q = \mathbf{0}$  by auto
          moreover have  $(\nu x)\mathbf{0} = (\nu y)\mathbf{0}$  by(subst alphaRes) auto

```

```

      ultimately show ?case using ⟨y # Ψ⟩ ⟨y # Ψ'⟩ by auto
    next
      assume (Ψ, P, Q) ∉ ?X1
      with ⟨(Ψ, P, Q) ∈ ?X⟩ have (Ψ, P, Q) ∈ ?X2 by auto
      then obtain x where Q = (νx)0 and P = 0 by auto
      moreover have (νx)0 = (νy) 0 by(subst alphaRes) auto
      ultimately show ?case using ⟨y # Ψ⟩ ⟨y # Ψ'⟩ by auto
    qed
  next
    case(cSym Ψ P Q)
    then show ?case by auto
  qed
}
moreover obtain y::name where y # Ψ by(generate-fresh name) auto
ultimately have Ψ ▷ (νy)0 ~ 0 by auto
then show ?thesis by(subst alphaRes[where x=x and y=y]) auto
qed

```

lemma *bisimOutputPushRes*:

```

fixes x :: name
and Ψ :: 'b
and M :: 'a
and N :: 'a
and P :: ('a, 'b, 'c) psi

```

```

assumes x # M
and x # N

```

shows $\Psi \triangleright (\nu x)(M\langle N \rangle.P) \sim M\langle N \rangle.(\nu x)P$

proof –

```

{
  fix x::name and P::('a, 'b, 'c) psi
  assume x # Ψ and x # M and x # N
  let ?X1 = {(Ψ, (νx)(M⟨N⟩.P), M⟨N⟩.(νx)P) | Ψ x M N P. x # Ψ ∧ x # M ∧
x # N}
  let ?X2 = {(Ψ, M⟨N⟩.(νx)P, (νx)(M⟨N⟩.P)) | Ψ x M N P. x # Ψ ∧ x # M ∧
x # N}
  let ?X = ?X1 ∪ ?X2

```

have *eqvt* ?X

by(rule *eqvtUnion*) (force simp add: *eqvt-def pt-fresh-bij*[OF *pt-name-inst*, OF *at-name-inst*] *eqvts*)+

from ⟨x # Ψ⟩ ⟨x # M⟩ ⟨x # N⟩ have (Ψ, (νx)(M⟨N⟩.P), M⟨N⟩.(νx)P) ∈ ?X
by auto

then have $\Psi \triangleright (\nu x)(M\langle N \rangle.P) \sim M\langle N \rangle.(\nu x)P$

proof(*coinduct* rule: *bisimCoinduct*)

case(*cStatEq* Ψ Q R)

then show ?case using *freshComp* by(force intro: *frameResFresh FrameS-*

```

tatEqSym)
next
  case(cSim Ψ Q R)
  then show ?case using ⟨eqvt ?X⟩
    by(auto intro!: outputPushResLeft outputPushResRight bisimReflexive)
next
  case(cExt Ψ Q R Ψ')
  show ?case
  proof(cases (Ψ, Q, R) ∈ ?X1)
    assume (Ψ, Q, R) ∈ ?X1
    then obtain x M N P where Qeq: Q = (νx)(M⟨N⟩.P) and Req: R =
M⟨N⟩.(νx)P and x # Ψ and x # M and x # N by auto
    obtain y::name where y # Ψ and y # Ψ' and y # M and y # N and y # P
    by(generate-fresh name) (auto simp add: fresh-prod)

    moreover then have (Ψ ⊗ Ψ', (νy)(M⟨N⟩.([(x, y)] · P)), M⟨N⟩.(νy)([(x,
y)] · P)) ∈ ?X by auto
    moreover from Qeq ⟨x # M⟩ ⟨y # M⟩ ⟨x # N⟩ ⟨y # N⟩ ⟨y # P⟩ have Q =
(νy)(M⟨N⟩.([(x, y)] · P))
    apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
    moreover from Req ⟨y # P⟩ have R = M⟨N⟩.(νy)([(x, y)] · P)
    apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
    ultimately show ?case by blast
  next
    assume (Ψ, Q, R) ∉ ?X1
    with ⟨(Ψ, Q, R) ∈ ?X⟩ have (Ψ, Q, R) ∈ ?X2 by blast
    then obtain x M N P where Req: R = (νx)(M⟨N⟩.P) and Qeq: Q =
M⟨N⟩.(νx)P and x # Ψ and x # M and x # N by auto
    obtain y::name where y # Ψ and y # Ψ' and y # M and y # N and y # P
    by(generate-fresh name) (auto simp add: fresh-prod)

    moreover then have (Ψ ⊗ Ψ', (νy)(M⟨N⟩.([(x, y)] · P)), M⟨N⟩.(νy)([(x,
y)] · P)) ∈ ?X by auto
    moreover from Req ⟨x # M⟩ ⟨y # M⟩ ⟨x # N⟩ ⟨y # N⟩ ⟨y # P⟩ have R =
(νy)(M⟨N⟩.([(x, y)] · P))
    apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
    moreover from Qeq ⟨y # P⟩ have Q = M⟨N⟩.(νy)([(x, y)] · P)
    apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
    ultimately show ?case by blast
  qed
next
  case(cSym Ψ R Q)
  then show ?case by blast
qed
}
moreover obtain y::name where y # Ψ and y # M and y # N y # P
by(generate-fresh name) auto
ultimately have Ψ ▷ (νy)(M⟨N⟩.([(x, y)] · P)) ∼ M⟨N⟩.(νy)([(x, y)] · P) by
auto

```

then show *?thesis* **using** *assms* $\langle y \# P \rangle \langle y \# M \rangle \langle y \# N \rangle$
apply(*subst alphaRes*[**where** $x=x$ **and** $y=y$ **and** $P=P$], *auto*)
by(*subst alphaRes*[**where** $x=x$ **and** $y=y$ **and** $P=M(N).P$]) *auto*
qed

lemma *bisimInputPushRes*:

fixes x **::** *name*
and Ψ **::** *'b*
and M **::** *'a*
and $xvec$ **::** *name list*
and N **::** *'a*
and P **::** (*'a, 'b, 'c*) *psi*

assumes $x \# M$
and $x \# xvec$
and $x \# N$

shows $\Psi \triangleright (\nu x)(M(\lambda * xvec N).P) \sim M(\lambda * xvec N).(\nu x)P$

proof –

{
fix $x::name$ **and** $P::('a, 'b, 'c)$ *psi*
assume $x \# \Psi$ **and** $x \# M$ **and** $x \# N$ **and** $x \# xvec$
let $?X1 = \{(\Psi, (\nu x)(M(\lambda * xvec N).P), M(\lambda * xvec N).(\nu x)P) \mid \Psi x M xvec N$
 $P. x \# \Psi \wedge x \# M \wedge x \# xvec \wedge x \# N\}$
let $?X2 = \{(\Psi, M(\lambda * xvec N).(\nu x)P, (\nu x)(M(\lambda * xvec N).P)) \mid \Psi x M xvec N$
 $P. x \# \Psi \wedge x \# M \wedge x \# xvec \wedge x \# N\}$
let $?X = ?X1 \cup ?X2$

have *eqvt* $?X$

by(*rule eqvtUnion*) (*force simp add: eqvt-def pt-fresh-bij[OF pt-name-inst, OF at-name-inst] eqvts*)**+**

from $\langle x \# \Psi \rangle \langle x \# M \rangle \langle x \# xvec \rangle \langle x \# N \rangle$ **have** $(\Psi, (\nu x)(M(\lambda * xvec N).P), M(\lambda * xvec N).(\nu x)P) \in ?X$

by *blast*

then have $\Psi \triangleright (\nu x)(M(\lambda * xvec N).P) \sim M(\lambda * xvec N).(\nu x)P$

proof(*coinduct rule: bisimCoinduct*)

case(*cStatEq* $\Psi Q R$)

then show *?case* **using** *freshComp* **by**(*force intro: frameResFresh FrameStatEqSym*)

next

case(*cSim* $\Psi Q R$)

then show *?case* **using** $\langle eqvt ?X \rangle$

by(*auto intro!: inputPushResLeft inputPushResRight bisimReflexive*)

next

case(*cExt* $\Psi Q R \Psi'$)

show *?case*

proof(*cases* $(\Psi, Q, R) \in ?X1$)

assume $(\Psi, Q, R) \in ?X1$

then obtain $x \# M \text{ xvec } N \ P$ **where** $Qeq: Q = (\nu x)(M(\lambda * \text{xvec } N).P)$ **and**
Req: $R = M(\lambda * \text{xvec } N).(\nu x)P$ **and** $x \# \Psi$
and $x \# M$ **and** $x \# \text{xvec}$ **and** $x \# N$ **by** *auto*
obtain $y::\text{name}$ **where** $y \# \Psi$ **and** $y \# \Psi'$ **and** $y \# M$ **and** $y \# N$ **and** $y \# P$
and $y \# \text{xvec}$
by(*generate-fresh name*) (*auto simp add: fresh-prod*)

moreover then have $(\Psi \otimes \Psi', (\nu y)(M(\lambda * \text{xvec } N).((x, y)) \cdot P)), M(\lambda * \text{xvec } N).(\nu y)((x, y) \cdot P) \in ?X$ **by** *force*
moreover from $Qeq \langle x \# M \rangle \langle y \# M \rangle \langle x \# \text{xvec} \rangle \langle y \# \text{xvec} \rangle \langle x \# N \rangle \langle y \# N \rangle \langle y \# P \rangle$ **have** $Q = (\nu y)(M(\lambda * \text{xvec } N).((x, y)) \cdot P)$
apply *clarsimp* **by**(*subst alphaRes[of y]*) (*auto simp add: eqvts inputChain-Fresh*)
moreover from *Req* $\langle y \# P \rangle$ **have** $R = M(\lambda * \text{xvec } N).(\nu y)((x, y) \cdot P)$
apply *clarsimp* **by**(*subst alphaRes[of y]*) (*auto simp add: eqvts*)
ultimately show *?case* **by** *blast*

next

assume $(\Psi, Q, R) \notin ?X1$
with $\langle \Psi, Q, R \rangle \in ?X$ **have** $(\Psi, Q, R) \in ?X2$ **by** *blast*
then obtain $x \# M \text{ xvec } N \ P$ **where** $Req: R = (\nu x)(M(\lambda * \text{xvec } N).P)$ **and**
Qeq: $Q = M(\lambda * \text{xvec } N).(\nu x)P$ **and** $x \# \Psi$
and $x \# M$ **and** $x \# \text{xvec}$ **and** $x \# N$ **by** *auto*
obtain $y::\text{name}$ **where** $y \# \Psi$ **and** $y \# \Psi'$ **and** $y \# M$ **and** $y \# N$ **and** $y \# P$
and $y \# \text{xvec}$
by(*generate-fresh name*) (*auto simp add: fresh-prod*)

moreover then have $(\Psi \otimes \Psi', (\nu y)(M(\lambda * \text{xvec } N).((x, y)) \cdot P)), M(\lambda * \text{xvec } N).(\nu y)((x, y) \cdot P) \in ?X$ **by** *force*
moreover from *Req* $\langle x \# M \rangle \langle y \# M \rangle \langle x \# \text{xvec} \rangle \langle y \# \text{xvec} \rangle \langle x \# N \rangle \langle y \# N \rangle \langle y \# P \rangle$ **have** $R = (\nu y)(M(\lambda * \text{xvec } N).((x, y)) \cdot P)$
apply *clarsimp* **by**(*subst alphaRes[of y]*) (*auto simp add: eqvts inputChain-Fresh*)
moreover from *Qeq* $\langle y \# P \rangle$ **have** $Q = M(\lambda * \text{xvec } N).(\nu y)((x, y) \cdot P)$
apply *clarsimp* **by**(*subst alphaRes[of y]*) (*auto simp add: eqvts*)
ultimately show *?case* **by** *blast*

qed

next

case(*cSym* $\Psi \ R \ Q$)
then show *?case* **by** *blast*

qed

}

moreover obtain $y::\text{name}$ **where** $y \# \Psi$ **and** $y \# M$ **and** $y \# N$ **and** $y \# P$ **and**
 $y \# \text{xvec}$
by(*generate-fresh name*) *auto*
ultimately have $\Psi \triangleright (\nu y)(M(\lambda * \text{xvec } N).((x, y)) \cdot P) \sim M(\lambda * \text{xvec } N).(\nu y)((x, y) \cdot P)$ **by** *auto*

then show *?thesis* **using** *assms* $\langle y \# P \rangle \langle y \# M \rangle \langle y \# N \rangle \langle y \# \text{xvec} \rangle$

apply(*subst alphaRes[where x=x and y=y and P=P]*, *auto*)

by(*subst alphaRes[where x=x and y=y and P=M(\lambda * \text{xvec } N).P]*) (*auto simp*)

add: inputChainFresh eqvts
qed

lemma *bisimCasePushRes:*

fixes $x :: \text{name}$
and $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c) \text{psi}) \text{list}$

assumes $x \# (\text{map fst } Cs)$

shows $\Psi \triangleright (\nu x)(\text{Cases } Cs) \sim \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs)$

proof –

{
fix $x :: \text{name}$ **and** $Cs :: ('c \times ('a, 'b, 'c) \text{psi}) \text{list}$
assume $x \# \Psi$ **and** $x \# (\text{map fst } Cs)$
let $?X1 = \{(\Psi, (\nu x)(\text{Cases } Cs), \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs)) \mid \Psi x$
 $Cs. x \# \Psi \wedge x \# (\text{map fst } Cs)\}$
let $?X2 = \{(\Psi, \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs), (\nu x)(\text{Cases } Cs)) \mid \Psi x$
 $Cs. x \# \Psi \wedge x \# (\text{map fst } Cs)\}$
let $?X = ?X1 \cup ?X2$

have *eqvt ?X*

proof

show *eqvt ?X1*

apply(*clarsimp simp add: eqvt-def eqvts*)
apply (*rule exI*)
apply (*rule exI*)
apply (*rule conjI*)
apply (*rule refl*)
apply(*perm-extend-simp*)
apply(*clarsimp simp add: eqvts*)
apply (*simp add: fresh-bij*)
apply(*drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst]*)
apply(*drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst]*)
apply(*simp add: eqvts*)
apply(*perm-extend-simp*)
apply(*simp add: eqvts*)
done

next

show *eqvt ?X2*

apply(*clarsimp simp add: eqvt-def eqvts*)
apply (*rule exI*)
apply (*rule exI*)
apply (*subst conj-commute*)
apply (*rule conjI*)
apply (*rule conjI*)
apply (*rule refl*)
apply(*perm-extend-simp*)
apply (*simp add: fresh-bij*)

```

apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
apply(drule pt-fresh-bij1[OF pt-name-inst, OF at-name-inst])
apply(simp add: eqvts)
apply(perm-extend-simp)
apply(simp add: eqvts)
apply(perm-extend-simp)
apply(clarsimp simp add: eqvts)
done
qed

from  $\langle x \# \Psi \rangle \langle x \# \text{map fst } Cs \rangle$  have  $(\Psi, (\nu x)(\text{Cases } Cs), \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P))) Cs)) \in ?X$  by auto
then have  $\Psi \triangleright (\nu x)(\text{Cases } Cs) \sim \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P))) Cs$ 
proof(coinduct rule: bisimCoinduct)
  case(cStatEq  $\Psi$   $Q$   $R$ )
    then show ?case using freshComp by(force intro: frameResFresh FrameStatEqSym)
  next
    case(cSim  $\Psi$   $Q$   $R$ )
    then show ?case using  $\langle \text{eqvt } ?X \rangle$ 
      by(auto intro!: casePushResLeft casePushResRight bisimReflexive)
  next
    case(cExt  $\Psi$   $Q$   $R$   $\Psi'$ )
    show ?case
    proof(cases  $(\Psi, Q, R) \in ?X1$ )
      assume  $(\Psi, Q, R) \in ?X1$ 
      then obtain  $x$   $Cs$  where  $Qeq: Q = (\nu x)(\text{Cases } Cs)$  and  $Req: R = \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P))) Cs$ 
        and  $x \# \Psi$  and  $x \# (\text{map fst } Cs)$  by blast
        obtain  $y::\text{name}$  where  $y \# \Psi$  and  $y \# \Psi'$  and  $y \# Cs$ 
          by(generate-fresh name) (auto simp add: fresh-prod)
          from  $\langle y \# Cs \rangle \langle x \# (\text{map fst } Cs) \rangle$  have  $y \# \text{map fst } ([ (x, y) ] \cdot Cs)$  by(induct  $Cs$ ) (auto simp add: fresh-list-cons fresh-list-nil)

      moreover with  $\langle y \# \Psi \rangle \langle y \# \Psi' \rangle$  have  $(\Psi \otimes \Psi', (\nu y)(\text{Cases } ([ (x, y) ] \cdot Cs)), \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu y)P))) ([ (x, y) ] \cdot Cs))) \in ?X$ 
        by auto
        moreover from  $Qeq \langle y \# Cs \rangle$  have  $Q = (\nu y)(\text{Cases } ([ (x, y) ] \cdot Cs))$ 
          apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
        moreover from  $Req \langle y \# Cs \rangle \langle x \# (\text{map fst } Cs) \rangle$  have  $R = \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu y)P))) ([ (x, y) ] \cdot Cs)$ 
          by(induct  $Cs$  arbitrary: R) (auto simp add: fresh-list-cons fresh-prod alphaRes)
        ultimately show ?case by blast
    next
      assume  $(\Psi, Q, R) \notin ?X1$ 
      with  $\langle (\Psi, Q, R) \in ?X \rangle$  have  $(\Psi, Q, R) \in ?X2$  by blast
      then obtain  $x$   $Cs$  where  $Req: R = (\nu x)(\text{Cases } Cs)$  and  $Qeq: Q = \text{Cases}(\text{map } (\lambda(\varphi, P). (\varphi, (\nu x)P))) Cs$ 

```

```

    and x # Ψ and x # (map fst Cs) by blast
    obtain y::name where y # Ψ and y # Ψ' and y # Cs
    by(generate-fresh name) (auto simp add: fresh-prod)
    from ⟨y # Cs⟩ ⟨x # (map fst Cs)⟩ have y # map fst ([x, y] · Cs) by(induct
Cs) (auto simp add: fresh-list-cons fresh-list-nil)

    moreover with ⟨y # Ψ⟩ ⟨y # Ψ'⟩ have (Ψ ⊗ Ψ', (νy)(Cases ([x, y] · Cs)),
Cases(map (λ(φ, P). (φ, (νy)P)) ([x, y] · Cs))) ∈ ?X
    by auto
    moreover from Req ⟨y # Cs⟩ have R = (νy)(Cases([x, y] · Cs))
    apply clarsimp by(subst alphaRes[of y]) (auto simp add: eqvts)
    moreover from Qeq ⟨y # Cs⟩ ⟨x # (map fst Cs)⟩ have Q = Cases(map
(λ(φ, P). (φ, (νy)P)) ([x, y] · Cs))
    by(induct Cs arbitrary: Q) (auto simp add: fresh-list-cons fresh-prod
alphaRes)
    ultimately show ?case by blast
  qed
next
  case(cSym Ψ R Q)
  then show ?case by blast
  qed
}
moreover obtain y::name where y # Ψ and y # Cs by(generate-fresh name)
auto
moreover from ⟨x # map fst Cs⟩ have y # map fst([x, y] · Cs)
  by(induct Cs) (auto simp add: fresh-left calc-atm)
ultimately have Ψ ▷ (νy)(Cases ([x, y] · Cs)) ~ Cases(map (λ(φ, P). (φ,
(νy)P)) ([x, y] · Cs))
  by auto
moreover from ⟨y # Cs⟩ have (νy)(Cases ([x, y] · Cs)) = (νx)(Cases Cs)
  by(simp add: alphaRes eqvts)
moreover from ⟨x # map fst Cs⟩ ⟨y # Cs⟩ have Cases(map (λ(φ, P). (φ, (νy)P))
([x, y] · Cs)) = Cases(map (λ(φ, P). (φ, (νx)P)) Cs)
  by(induct Cs) (auto simp add: alphaRes)
ultimately show ?thesis by auto
qed

lemma bangExt:
  fixes Ψ :: 'b
  and P :: ('a, 'b, 'c) psi

assumes guarded P

shows Ψ ▷ !P ~ P || !P
proof -
  let ?X = {(Ψ, !P, P || !P) | Ψ P. guarded P} ∪ {(Ψ, P || !P, !P) | Ψ P. guarded
P}
  from ⟨guarded P⟩ have (Ψ, !P, P || !P) ∈ ?X by auto
  then show ?thesis

```

```

proof(coinduct rule: bisimCoinduct)
  case(cStatEq  $\Psi$   $Q$   $R$ )
    from  $\langle (\Psi, Q, R) \in ?X \rangle$  obtain  $P$  where  $Eq: (Q = !P \wedge R = P \parallel !P) \vee (Q$ 
  =  $P \parallel !P \wedge R = !P)$  and guarded  $P$ 
    by auto
    obtain  $A_P$   $\Psi_P$  where  $FrP: extractFrame P = \langle A_P, \Psi_P \rangle$  and  $A_P \#* \Psi$  by(rule
freshFrame)
    from  $FrP$   $\langle guarded P \rangle$  have  $\Psi_P \simeq SBottom'$  by(blast dest: guardedStatEq)
    from  $\langle \Psi_P \simeq SBottom' \rangle$  have  $\Psi \otimes SBottom' \simeq \Psi \otimes \Psi_P \otimes SBottom'$  by(metis
Identity Composition AssertionStatEqTrans Commutativity AssertionStatEqSym)
    then have  $\langle A_P, \Psi \otimes SBottom' \rangle \simeq_F \langle A_P, \Psi \otimes \Psi_P \otimes SBottom' \rangle$ 
    by(force intro: frameResChainPres)
    moreover from  $\langle A_P \#* \Psi \rangle$  have  $\langle \varepsilon, \Psi \otimes SBottom' \rangle \simeq_F \langle A_P, \Psi \otimes SBottom' \rangle$ 
    apply –
    by(rule FrameStatEqSym) (simp add: frameResFreshChain freshCompChain(1))
    ultimately show  $?case$  using  $Eq \langle A_P \#* \Psi \rangle FrP$ 
    by auto (blast dest: FrameStatEqTrans FrameStatEqSym)+
  next
    case(cSim  $\Psi$   $Q$   $R$ )
    then show  $?case$  by(auto intro: bangExtLeft bangExtRight bisimReflexive)
  next
    case(cExt  $\Psi$   $Q$   $R$ )
    then show  $?case$  by auto
  next
    case(cSym  $\Psi$   $Q$   $R$ )
    then show  $?case$  by auto
  qed
qed

```

lemma *bisimScopeExtSym:*

```

fixes  $x :: name$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $P :: ('a, 'b, 'c) psi$ 

```

```

assumes  $x \# \Psi$ 
  and  $x \# Q$ 

```

shows $\Psi \triangleright (\nu x)(P \parallel Q) \sim ((\nu x)P) \parallel Q$

```

using assms
  by(metis bisimScopeExt bisimTransitive bisimParComm bisimSymmetric bisim-
ResPres)

```

lemma *bisimScopeExtChainSym:*

```

fixes  $xvec :: name list$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $P :: ('a, 'b, 'c) psi$ 

```

```

assumes  $xvec \#* \Psi$ 
  and  $xvec \#* Q$ 

```

shows $\Psi \triangleright (\nu^*xvec)(P \parallel Q) \sim ((\nu^*xvec)P) \parallel Q$
using *assms*
by(*induct xvec*) (*auto intro: bisimScopeExtSym bisimReflexive bisimTransitive bisimResPres*)

lemma *bisimParPresAuxSym*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $Q :: ('a, 'b, 'c) psi$
and $R :: ('a, 'b, 'c) psi$

assumes $\Psi \otimes \Psi_R \triangleright P \sim Q$
and *extractFrame* $R = \langle A_R, \Psi_R \rangle$
and $A_R \#* \Psi$
and $A_R \#* P$
and $A_R \#* Q$

shows $\Psi \triangleright R \parallel P \sim R \parallel Q$
using *assms*
by(*metis bisimParComm bisimParPresAux bisimTransitive*)

lemma *bangDerivative*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c) psi$
and $\alpha :: 'a action$
and $P' :: ('a, 'b, 'c) psi$

assumes $\Psi \triangleright !P \mapsto \alpha \prec P'$
and $\Psi \triangleright P \sim Q$
and $bn \alpha \#* \Psi$
and $bn \alpha \#* P$
and $bn \alpha \#* Q$
and $bn \alpha \#* subject \alpha$
and *guarded* Q

obtains $Q' R T$ **where** $\Psi \triangleright !Q \mapsto \alpha \prec Q'$ **and** $\Psi \triangleright P' \sim R \parallel !P$ **and** $\Psi \triangleright Q' \sim T \parallel !Q$ **and** $\Psi \triangleright R \sim T$
and $((supp R)::name set) \subseteq supp P'$ **and** $((supp T)::name set) \subseteq supp Q'$

proof –

from $\langle \Psi \triangleright !P \mapsto \alpha \prec P' \rangle$ **have** *guarded* P

apply –

by(*ind-cases* $\Psi \triangleright !P \mapsto \alpha \prec P'$) (*auto simp add: psi.inject*)

assume $\bigwedge Q' R T. \llbracket \Psi \triangleright !Q \mapsto \alpha \prec Q'; \Psi \triangleright P' \sim R \parallel !P; \Psi \triangleright Q' \sim T \parallel !Q; \Psi \triangleright R \sim T; ((supp R)::name set) \subseteq supp P';$

$((supp T)::name set) \subseteq supp Q' \rrbracket \implies thesis$

moreover from $\langle \Psi \triangleright !P \mapsto \alpha \prec P' \rangle \langle bn \alpha \#* subject \alpha \rangle \langle bn \alpha \#* \Psi \rangle \langle bn \alpha \#* P \rangle \langle bn \alpha \#* Q \rangle \langle \Psi \triangleright P \sim Q \rangle \langle guarded Q \rangle$

have $\exists Q' T R. \Psi \triangleright !Q \mapsto \alpha \prec Q' \wedge \Psi \triangleright P' \sim R \parallel !P \wedge \Psi \triangleright Q' \sim T \parallel !Q$

$\wedge \Psi \triangleright R \sim T \wedge$
 $((\text{supp } R)::\text{name set}) \subseteq \text{supp } P' \wedge ((\text{supp } T)::\text{name set}) \subseteq \text{supp } Q'$
proof(*nominal-induct avoiding: Q rule: bangInduct'*)
case(*cAlpha* α $P' p$ Q)
then obtain $Q' T R$ **where** $QTrans: \Psi \triangleright !Q \mapsto \alpha \prec Q'$ **and** $\Psi \triangleright P' \sim R \parallel$
 $(P \parallel !P)$ **and** $\Psi \triangleright Q' \sim T \parallel !Q$ **and** $\Psi \triangleright R \sim T$
and $\text{supp}R: ((\text{supp } R)::\text{name set}) \subseteq \text{supp } P'$ **and** $\text{supp}T: ((\text{supp } T)::\text{name set}) \subseteq \text{supp } Q'$
by *blast*
from $QTrans$ **have** $\text{distinct}(bn \ \alpha)$
by(*rule boundOutputDistinct*)
have $S: \text{set } p \subseteq \text{set}(bn \ \alpha) \times \text{set}(bn(p \cdot \alpha))$
by *fact*
from $QTrans \langle bn(p \cdot \alpha) \#* Q \rangle \langle bn(p \cdot \alpha) \#* \alpha \rangle \langle bn \ \alpha \#* \text{subject } \alpha \rangle \langle \text{distinct}(bn \ \alpha) \rangle$
have $bn(p \cdot \alpha) \#* Q'$
by(*drule-tac freeFreshChainDerivative*) *simp+*
with $QTrans \langle bn(p \cdot \alpha) \#* \alpha \rangle S \langle bn \ \alpha \#* \text{subject } \alpha \rangle$ **have** $\Psi \triangleright !Q \mapsto (p \cdot \alpha) \prec$
 $(p \cdot Q')$
by(*force simp add: residualAlpha*)
moreover from $\langle \Psi \triangleright P' \sim R \parallel (P \parallel !P) \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot P') \sim (p \cdot (R \parallel (P \parallel !P)))$
by(*rule bisimClosed*)
with $\langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* P \rangle \langle bn(p \cdot \alpha) \#* \Psi \rangle \langle bn(p \cdot \alpha) \#* P \rangle S$ **have** $\Psi \triangleright$
 $(p \cdot P') \sim (p \cdot R) \parallel (P \parallel !P)$
by(*simp add: eqts*)
moreover from $\langle \Psi \triangleright Q' \sim T \parallel !Q \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot Q') \sim (p \cdot (T \parallel !Q))$
by(*rule bisimClosed*)
with $\langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* Q \rangle \langle bn(p \cdot \alpha) \#* \Psi \rangle \langle bn(p \cdot \alpha) \#* Q \rangle S$ **have** $\Psi \triangleright$
 $(p \cdot Q') \sim (p \cdot T) \parallel !Q$
by(*simp add: eqts*)
moreover from $\langle \Psi \triangleright R \sim T \rangle$ **have** $(p \cdot \Psi) \triangleright (p \cdot R) \sim (p \cdot T)$
by(*rule bisimClosed*)
with $\langle bn \ \alpha \#* \Psi \rangle \langle bn(p \cdot \alpha) \#* \Psi \rangle S$ **have** $\Psi \triangleright (p \cdot R) \sim (p \cdot T)$
by(*simp add: eqts*)
moreover from $\text{supp}R$ **have** $((\text{supp}(p \cdot R))::\text{name set}) \subseteq \text{supp}(p \cdot P')$
apply(*erule-tac rev-mp*)
by(*subst subsetClosed[of p, symmetric]*) (*simp add: eqts*)
moreover from $\text{supp}T$ **have** $((\text{supp}(p \cdot T))::\text{name set}) \subseteq \text{supp}(p \cdot Q')$
apply(*erule-tac rev-mp*)
by(*subst subsetClosed[of p, symmetric]*) (*simp add: eqts*)
ultimately show *?case* **by** *blast*
next
case(*cPar1* α $P' Q$)
from $\langle \Psi \triangleright P \sim Q \rangle \langle \Psi \triangleright P \mapsto \alpha \prec P' \rangle \langle bn \ \alpha \#* \Psi \rangle \langle bn \ \alpha \#* Q \rangle$
obtain Q' **where** $QTrans: \Psi \triangleright Q \mapsto \alpha \prec Q'$ **and** $\Psi \triangleright P' \sim Q'$
by(*blast dest: bisimE simE*)
from $QTrans$ **have** $\Psi \otimes SBottom' \triangleright Q \mapsto \alpha \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
then have $\Psi \triangleright Q \parallel !Q \mapsto \alpha \prec (Q' \parallel !Q)$

```

    using ⟨bn α #* Q⟩ by(intro Par1) (assumption | simp)+
  then have Ψ ▷ !Q ⟶α < (Q' || !Q)
    using ⟨guarded Q⟩ by(rule Bang)
  moreover from ⟨guarded P⟩ have Ψ ▷ P' || !P ~ P' || (P || !P)
    by(metis bangExt bisimParPresSym)
  moreover have Ψ ▷ Q' || !Q ~ Q' || !Q
    by(rule bisimReflexive)
  ultimately show ?case using ⟨Ψ ▷ P' ~ Q'⟩
    by(force simp add: psi.supp)
next
  case(cPar2 α P' Q)
  then obtain Q' T R where QTrans: Ψ ▷ !Q ⟶α < Q' and Ψ ▷ P' ~ R ||
!P and Ψ ▷ Q' ~ T || !Q
    and Ψ ▷ R ~ T and suppR: ((supp R)::name set) ⊆ supp P' and suppT:
((supp T)::name set) ⊆ supp Q'
    by blast
  note QTrans
  from ⟨Ψ ▷ P' ~ R || !P⟩ have Ψ ▷ P || P' ~ R || (P || !P)
    by(metis bisimParPresSym bisimParComm bisimTransitive bisimParAssoc)
  with QTrans show ?case using ⟨Ψ ▷ Q' ~ T || !Q⟩ ⟨Ψ ▷ R ~ T⟩ suppR
suppT
    by(force simp add: psi.supp)
next
  case(cComm1 M N P' K xvec P'' Q)
  from ⟨Ψ ▷ P ~ Q⟩ have Ψ ▷ Q ~[bisim] P
    by(metis bisimE)
  with ⟨Ψ ▷ P ⟶M(|N) < P'⟩ obtain Q' where QTrans: Ψ ▷ Q ⟶M(|N)
< Q' and Ψ ▷ Q' ~ P'
    by(force dest: simE)
  from QTrans have Ψ ⊗ SBottom' ▷ Q ⟶M(|N) < Q'
    by(metis statEqTransition Identity AssertionStatEqSym)
  moreover obtain A_Q Ψ_Q where FrQ: extractFrame Q = ⟨A_Q, Ψ_Q⟩ and A_Q
#* Ψ and A_Q #* Q and A_Q #* M
    by(rule freshFrame[where C=(Ψ, Q, M)]) auto
  note FrQ
  moreover from FrQ ⟨guarded Q⟩ have Ψ_Q ≅ SBottom'
    by(blast dest: guardedStatEq)
  from ⟨Ψ ▷ !P ⟶K(|ν*xvec)⟨N⟩ < P''⟩ ⟨xvec #* K⟩ ⟨Ψ ▷ P ~ Q⟩ ⟨xvec #*
Ψ⟩ ⟨xvec #* P⟩ ⟨xvec #* Q⟩ ⟨guarded Q⟩
    obtain Q'' T R where QTrans': Ψ ▷ !Q ⟶K(|ν*xvec)⟨N⟩ < Q'' and Ψ ▷
P'' ~ R || !P and Ψ ▷ Q'' ~ T || !Q and Ψ ▷ R ~ T
    and suppR: ((supp R)::name set) ⊆ supp P'' and suppT: ((supp T)::name
set) ⊆ supp Q''
    using cComm1 by force
  from QTrans' ⟨Ψ_Q ≅ SBottom'⟩ have Ψ ⊗ Ψ_Q ▷ !Q ⟶K(|ν*xvec)⟨N⟩ < Q''
    by(metis statEqTransition Identity compositionSym AssertionStatEqSym)
  moreover from ⟨Ψ ⊢ M ↔ K⟩ ⟨Ψ_Q ≅ SBottom'⟩ have Ψ ⊗ Ψ_Q ⊂ SBottom'
⊢ M ↔ K
    by(metis statEqEnt Identity compositionSym AssertionStatEqSym)

```

```

ultimately have  $\Psi \triangleright Q \parallel !Q \mapsto \tau \prec ((\nu * xvec)(Q' \parallel Q''))$  using  $\langle A_Q \#* \Psi \rangle$ 
 $\langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle xvec \#* Q \rangle$ 
  by(intro Comm1) (assumption | simp)+
then have  $\Psi \triangleright !Q \mapsto \tau \prec ((\nu * xvec)(Q' \parallel Q''))$ 
  using  $\langle guarded Q \rangle$  by(rule Bang)
  moreover from  $\langle \Psi \triangleright P'' \sim R \parallel !P \rangle \langle guarded P \rangle \langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright$ 
 $(\nu * xvec)(P' \parallel P'') \sim (\nu * xvec)((P' \parallel R) \parallel (P \parallel !P))$ 
  by(metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSym-
metric bisimResChainPres)
  with  $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle$  have  $\Psi \triangleright (\nu * xvec)(P' \parallel P'') \sim ((\nu * xvec)(P' \parallel$ 
 $R)) \parallel (P \parallel !P)$ 
  by(metis bisimScopeExtChainSym bisimTransitive psiFreshVec)
  moreover from  $\langle \Psi \triangleright Q'' \sim T \parallel !Q \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* Q \rangle$  have  $\Psi \triangleright$ 
 $(\nu * xvec)(Q' \parallel Q'') \sim ((\nu * xvec)(Q' \parallel T)) \parallel !Q$ 
  by(metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric
bisimResChainPres bisimScopeExtChainSym psiFreshVec)
  moreover from  $\langle \Psi \triangleright R \sim T \rangle \langle \Psi \triangleright Q' \sim P' \rangle \langle xvec \#* \Psi \rangle$  have  $\Psi \triangleright (\nu * xvec)(P'$ 
 $\parallel R) \sim (\nu * xvec)(Q' \parallel T)$ 
  by(metis bisimParPresSym bisimTransitive bisimResChainPres bisimParComm
bisimE(4))
  moreover from suppR have  $((supp((\nu * xvec)(P' \parallel R)))::name set) \subseteq supp(((\nu * xvec)(P'$ 
 $\parallel P'')))$ 
  by(auto simp add: psi.supp resChainSupp)
  moreover from suppT have  $((supp((\nu * xvec)(Q' \parallel T)))::name set) \subseteq supp(((\nu * xvec)(Q'$ 
 $\parallel Q'')))$ 
  by(auto simp add: psi.supp resChainSupp)
ultimately show ?case by blast
next
case(cComm2 M xvec N P' K P'' Q)
from  $\langle \Psi \triangleright P \sim Q \rangle \langle \Psi \triangleright P \mapsto M(\nu * xvec)\langle N \rangle \prec P' \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* Q \rangle$ 
obtain Q' where QTrans:  $\Psi \triangleright Q \mapsto M(\nu * xvec)\langle N \rangle \prec Q'$  and  $\Psi \triangleright P' \sim Q'$ 
  by(metis bisimE simE bn.simps)
from QTrans have  $\Psi \otimes SBottom' \triangleright Q \mapsto M(\nu * xvec)\langle N \rangle \prec Q'$ 
  by(metis statEqTransition Identity AssertionStatEqSym)
moreover obtain  $A_Q \Psi_Q$  where FrQ: extractFrame  $Q = \langle A_Q, \Psi_Q \rangle$  and  $A_Q$ 
 $\#* \Psi$  and  $A_Q \#* Q$  and  $A_Q \#* M$ 
  by(rule freshFrame[where  $C=(\Psi, Q, M)$ ]) auto
note FrQ
moreover from FrQ  $\langle guarded Q \rangle$  have  $\Psi_Q \simeq SBottom'$ 
  by(blast dest: guardedStatEq)
from  $\langle \Psi \triangleright !P \mapsto K\langle N \rangle \prec P'' \rangle \langle \Psi \triangleright P \sim Q \rangle \langle guarded Q \rangle$ 
obtain  $Q'' T R$  where QTrans':  $\Psi \triangleright !Q \mapsto K\langle N \rangle \prec Q''$  and  $\Psi \triangleright P'' \sim R$ 
 $\parallel !P$  and  $\Psi \triangleright Q'' \sim T \parallel !Q$  and  $\Psi \triangleright R \sim T$ 
  and suppR:  $((supp R)::name set) \subseteq supp P''$  and suppT:  $((supp T)::name$ 
 $set) \subseteq supp Q''$  using cComm2
  by force
from QTrans'  $\langle \Psi_Q \simeq SBottom' \rangle$  have  $\Psi \otimes \Psi_Q \triangleright !Q \mapsto K\langle N \rangle \prec Q''$ 
  by(metis statEqTransition Identity compositionSym AssertionStatEqSym)
moreover from  $\langle \Psi \vdash M \leftrightarrow K \rangle \langle \Psi_Q \simeq SBottom' \rangle$  have  $\Psi \otimes \Psi_Q \otimes SBottom'$ 

```


$\vdash M \leftrightarrow K$
by(*metis statEqEnt Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \triangleright Q \parallel !Q \mapsto \tau \prec ((\nu * xvec)(Q' \parallel Q''))$ **using** $\langle A_Q \# * \Psi \rangle$
 $\langle A_Q \# * Q \rangle \langle A_Q \# * M \rangle \langle xvec \# * Q \rangle$
by(*intro Comm2*) (*assumption* | *simp*)+
then have $\Psi \triangleright !Q \mapsto \tau \prec ((\nu * xvec)(Q' \parallel Q''))$ **using** $\langle guarded Q \rangle$ **by**(*rule Bang*)
moreover from $\langle \Psi \triangleright P'' \sim R \parallel !P \rangle \langle guarded P \rangle \langle xvec \# * \Psi \rangle$ **have** $\Psi \triangleright$
 $(\nu * xvec)(P' \parallel P'') \sim (\nu * xvec)((P' \parallel R) \parallel (P \parallel !P))$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres*)
with $\langle xvec \# * \Psi \rangle \langle xvec \# * P \rangle$ **have** $\Psi \triangleright (\nu * xvec)(P' \parallel P'') \sim ((\nu * xvec)(P' \parallel R)) \parallel (P \parallel !P)$
by(*metis bisimScopeExtChainSym bisimTransitive psiFreshVec*)
moreover from $\langle \Psi \triangleright Q'' \sim T \parallel !Q \rangle \langle xvec \# * \Psi \rangle \langle xvec \# * Q \rangle$ **have** $\Psi \triangleright$
 $(\nu * xvec)(Q' \parallel Q'') \sim ((\nu * xvec)(Q' \parallel T)) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric bisimResChainPres bisimScopeExtChainSym psiFreshVec*)
moreover from $\langle \Psi \triangleright R \sim T \rangle \langle \Psi \triangleright P' \sim Q' \rangle \langle xvec \# * \Psi \rangle$ **have** $\Psi \triangleright (\nu * xvec)(P' \parallel R) \sim (\nu * xvec)(Q' \parallel T)$
by(*metis bisimParPresSym bisimTransitive bisimResChainPres bisimParComm*)
moreover from *suppR* **have** $((supp((\nu * xvec)(P' \parallel R)))::name set) \subseteq supp((\nu * xvec)(P' \parallel P''))$
by(*auto simp add: psi.supp resChainSupp*)
moreover from *suppT* **have** $((supp((\nu * xvec)(Q' \parallel T)))::name set) \subseteq supp((\nu * xvec)(Q' \parallel Q''))$
by(*auto simp add: psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
next
case(*cBang* $\alpha P' Q$)
then obtain $Q' T R$ **where** $QTrans: \Psi \triangleright !Q \mapsto \alpha \prec Q'$ **and** $\Psi \triangleright P' \sim R \parallel (P \parallel !P)$ **and** $\Psi \triangleright Q' \sim T \parallel !Q$ **and** $\Psi \triangleright R \sim T$
and *suppR*: $((supp R)::name set) \subseteq supp P'$ **and** *suppT*: $((supp T)::name set) \subseteq supp Q'$
by *blast*
from $\langle \Psi \triangleright P' \sim R \parallel (P \parallel !P) \rangle \langle guarded P \rangle$ **have** $\Psi \triangleright P' \sim R \parallel !P$
by(*metis bangExt bisimParPresSym bisimTransitive bisimSymmetric*)
with $QTrans$ **show** *?case* **using** $\langle \Psi \triangleright Q' \sim T \parallel !Q \rangle \langle \Psi \triangleright R \sim T \rangle$ *suppR* *suppT*
by *blast*
next
case(*cBrMerge* $M N P' P'' Q$)
then obtain $Q' T R$ **where** $\Psi \triangleright !Q \mapsto iM(N) \prec Q'$ **and**
 $p''eq: \Psi \triangleright P'' \sim R \parallel !P$ **and**
 $\Psi \triangleright Q' \sim T \parallel !Q$ **and** $\Psi \triangleright R \sim T$ **and**
 $supp R \subseteq (supp P'':name set)$ **and**
 $supp T \subseteq (supp Q':name set)$
by *blast*

```

obtain  $Q''$  where  $\Psi \triangleright Q \mapsto iM(N) \prec Q''$  and  $\Psi \triangleright Q'' \sim P'$ 
proof –
  assume  $g: (\bigwedge Q''. [\Psi \triangleright Q \mapsto iM(N) \prec Q''; \Psi \triangleright Q'' \sim P'] \implies thesis)$ 
  have  $\exists Q'. \Psi \triangleright Q \mapsto iM(N) \prec Q' \wedge (\Psi, Q', P') \in bisim$ 
    apply(rule simE)
    apply(rule bisimE)
    apply(rule bisimSymmetric)
    by fact+
  then show thesis
    using  $g$  by blast
qed
have  $\Psi \otimes \mathbf{1} \triangleright Q \mapsto iM(N) \prec Q''$  using  $\langle \Psi \triangleright Q \mapsto iM(N) \prec Q'' \rangle$ 
  by(rule statEqTransition) (rule AssertionStatEqSym[OF Identity])
obtain  $A_Q \Psi_Q$  where  $extractFrame Q = \langle A_Q, \Psi_Q \rangle$  and  $distinct A_Q$  and  $A_Q$ 
 $\#* \Psi$  and  $A_Q \#* Q$  and  $A_Q \#* M$ 
  by(rule freshFrame[where  $C=(\Psi, Q, M)$ ]) force
then have  $\Psi_Q \simeq \mathbf{1}$  using  $\langle guarded Q \rangle$ 
  by(auto dest: guardedStatEq)
have  $\Psi \otimes \Psi_Q \triangleright !Q \mapsto iM(N) \prec Q'$  using  $\langle \Psi \triangleright !Q \mapsto iM(N) \prec Q' \rangle$ 
  by(rule statEqTransition) (metis  $\langle \Psi_Q \simeq \mathbf{1} \rangle$  AssertionStatEqTrans Assertion-
StatEqSym Identity compositionSym)
have  $\Psi \triangleright !Q \mapsto iM(N) \prec Q'' \parallel Q'$ 
  by(rule Bang) (rule BrMerge|fact|simp)+
moreover have  $\Psi \triangleright P' \parallel P'' \sim (P' \parallel R) \parallel (P \parallel !P)$ 
  apply(rule bisimTransitive[OF bisimParPresSym, OF p''eq])
  apply(rule bisimTransitive[OF bisimSymmetric, OF bisimParAssoc])
  apply(rule bisimParPresSym)
  apply(rule bangExt[OF  $\langle guarded P \rangle$ ])
  done
moreover have  $\Psi \triangleright Q'' \parallel Q' \sim Q'' \parallel T \parallel !Q$  using  $\langle \Psi \triangleright Q' \sim T \parallel !Q \rangle$ 
  by(metis bisimTransitive bisimParAssoc bisimParComm bisimParPres bisim-
Symmetric)
moreover have  $\Psi \triangleright (P' \parallel R) \sim Q'' \parallel T$  using  $\langle \Psi \triangleright R \sim T \rangle$  and  $\langle \Psi \triangleright Q'' \sim P' \rangle$ 
  apply –
  apply(erule bisimTransitive[OF bisimParPres, OF bisimSymmetric])
  apply(erule bisimTransitive[OF bisimParPresSym])
  by(rule bisimReflexive)
moreover have  $supp(P' \parallel R) \subseteq (supp(P' \parallel P''))::name\ set$ 
  using  $\langle supp R \subseteq (supp P'')::name\ set \rangle$  by(auto simp add: psi.supp)
moreover have  $supp(Q'' \parallel T) \subseteq (supp(Q'' \parallel Q'))::name\ set$ 
  using  $\langle supp T \subseteq (supp Q')::name\ set \rangle$  by(auto simp add: psi.supp)
ultimately show ?case
  by blast
next
case(cBrComm1  $M N P' xvec P'' Q$ )
from  $\langle \Psi \triangleright P \sim Q \rangle$  have  $\Psi \triangleright Q \rightsquigarrow[bisim] P$  by(metis bisimE)
with  $\langle \Psi \triangleright P \mapsto iM(N) \prec P' \rangle$  obtain  $Q'$  where  $QTrans: \Psi \triangleright Q \mapsto iM(N) \prec Q'$  and  $\Psi \triangleright Q' \sim P'$ 

```

by(*force dest: simE*)
from $QTrans$ **have** $\Psi \otimes SBottom' \triangleright Q \mapsto_i M(N) \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain $A_Q \Psi_Q$ **where** FrQ : $extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** A_Q
 $\#* \Psi$ **and** $A_Q \#* Q$ **and** $A_Q \#* M$
by(*rule freshFrame[where C=(Ψ, Q, M)] auto*)
note FrQ
moreover from $FrQ \langle guarded Q \rangle$ **have** $\Psi_Q \simeq SBottom'$
by(*blast dest: guardedStatEq*)
from $\langle \Psi \triangleright !P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P'' \rangle \langle \Psi \triangleright P \sim Q \rangle \langle xvec \#* \Psi \rangle \langle xvec \#*$
 $P \rangle \langle xvec \#* Q \rangle \langle guarded Q \rangle$
obtain $Q'' T R$ **where** $QTrans'$: $\Psi \triangleright !Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q''$ **and** $\Psi \triangleright$
 $P'' \sim R \parallel !P$ **and** $\Psi \triangleright Q'' \sim T \parallel !Q$
and $\Psi \triangleright R \sim T$ **and** $suppR$: $((supp R)::name set) \subseteq supp P''$ **and** $suppT$:
 $((supp T)::name set) \subseteq supp Q''$
using *cBrComm1* **by** *force*
from $QTrans' \langle \Psi_Q \simeq SBottom' \rangle$ **have** $\Psi \otimes \Psi_Q \triangleright !Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec$
 Q''
by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \triangleright Q \parallel !Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q' \parallel Q''$
using $\langle A_Q \#* \Psi \rangle \langle A_Q \#* Q \rangle \langle A_Q \#* M \rangle \langle xvec \#* Q \rangle$ **by**(*intro BrComm1*)
(*assumption | simp*)+
then have $\Psi \triangleright !Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q' \parallel Q''$
using $\langle guarded Q \rangle$ **by**(*rule Bang*)
moreover from $\langle \Psi \triangleright P'' \sim R \parallel !P \rangle \langle guarded P \rangle \langle xvec \#* \Psi \rangle$ **have** $\Psi \triangleright P' \parallel$
 $P'' \sim (P' \parallel R) \parallel (P \parallel !P)$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSym-*
metric)
moreover from $\langle \Psi \triangleright Q'' \sim T \parallel !Q \rangle \langle xvec \#* \Psi \rangle \langle xvec \#* Q \rangle$ **have** $\Psi \triangleright Q' \parallel$
 $Q'' \sim (Q' \parallel T) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from $\langle \Psi \triangleright R \sim T \rangle \langle \Psi \triangleright Q' \sim P' \rangle \langle xvec \#* \Psi \rangle$ **have** $\Psi \triangleright P' \parallel R$
 $\sim Q' \parallel T$
by(*metis bisimParPresSym bisimTransitive bisimParComm bisimE(4)*)
moreover from $suppR$ **have** $((supp(P' \parallel R))::name set) \subseteq supp((P' \parallel P''))$
by(*auto simp add: psi.supp resChainSupp*)
moreover from $suppT$ **have** $((supp(Q' \parallel T))::name set) \subseteq supp((Q' \parallel Q''))$
by(*auto simp add: psi.supp resChainSupp*)
ultimately show *?case* **by** *blast*
next
case(*cBrComm2 M N P' xvec P'' Q*)
from $\langle \Psi \triangleright P \sim Q \rangle$ **have** $\Psi \triangleright Q \rightsquigarrow[bisim] P$ **by**(*metis bisimE*)
with $\langle \Psi \triangleright P \mapsto_i M(\nu*xvec)\langle N \rangle \prec P' \rangle$ **obtain** Q' **where** $QTrans$: $\Psi \triangleright Q$
 $\mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'$ **and** $\Psi \triangleright Q' \sim P'$
using $\langle xvec \#* \Psi \rangle \langle xvec \#* P \rangle \langle xvec \#* Q \rangle$ **by**(*auto dest: simE*)
from $QTrans$ **have** $\Psi \otimes SBottom' \triangleright Q \mapsto_i M(\nu*xvec)\langle N \rangle \prec Q'$
by(*metis statEqTransition Identity AssertionStatEqSym*)
moreover obtain $A_Q \Psi_Q$ **where** FrQ : $extractFrame Q = \langle A_Q, \Psi_Q \rangle$ **and** A_Q
 $\#* \Psi$ **and** $A_Q \#* Q$ **and** $A_Q \#* M$

by(*rule freshFrame*[**where** $C=(\Psi, Q, M)$]) *auto*
note *FrQ*
moreover from *FrQ* \langle *guarded Q* \rangle **have** $\Psi_Q \simeq SBottom'$ **by**(*blast dest: guardedStatEq*)
from \langle $\Psi \triangleright !P \mapsto_i M(N) \prec P''$ \rangle \langle $\Psi \triangleright P \sim Q$ \rangle \langle *xvec #* Psi* \rangle \langle *xvec #* P* \rangle \langle *xvec #* Q* \rangle \langle *guarded Q* \rangle
obtain $Q'' T R$ **where** $QTrans': \Psi \triangleright !Q \mapsto_i M(N) \prec Q''$ **and** $\Psi \triangleright P'' \sim R$
 $\parallel !P$ **and** $\Psi \triangleright Q'' \sim T \parallel !Q$
and $\Psi \triangleright R \sim T$ **and** $suppR: ((supp R)::name set) \subseteq supp P''$ **and** $suppT: ((supp T)::name set) \subseteq supp Q''$
using *cBrComm2* **by force**
from $QTrans'$ \langle $\Psi_Q \simeq SBottom'$ \rangle **have** $\Psi \otimes \Psi_Q \triangleright !Q \mapsto_i M(N) \prec Q''$
by(*metis statEqTransition Identity compositionSym AssertionStatEqSym*)
ultimately have $\Psi \triangleright Q \parallel !Q \mapsto_i M(\nu*xvec)(N) \prec Q' \parallel Q''$
using \langle $A_Q \#* \Psi$ \rangle \langle $A_Q \#* Q$ \rangle \langle $A_Q \#* M$ \rangle \langle *xvec #* Q* \rangle **by**(*intro BrComm2*)
(*assumption* | *simp*)+
then have $\Psi \triangleright !Q \mapsto_i M(\nu*xvec)(N) \prec Q' \parallel Q''$
using \langle *guarded Q* \rangle **by**(*rule Bang*)
moreover from \langle $\Psi \triangleright P'' \sim R \parallel !P$ \rangle \langle *guarded P* \rangle \langle *xvec #* Psi* \rangle **have** $\Psi \triangleright P' \parallel P'' \sim (P' \parallel R) \parallel (P \parallel !P)$
by(*metis bisimParPresSym bangExt bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from \langle $\Psi \triangleright Q'' \sim T \parallel !Q$ \rangle \langle *xvec #* Psi* \rangle \langle *xvec #* Q* \rangle **have** $\Psi \triangleright Q' \parallel Q'' \sim (Q' \parallel T) \parallel !Q$
by(*metis bisimParPresSym bisimTransitive bisimParAssoc bisimSymmetric*)
moreover from \langle $\Psi \triangleright R \sim T$ \rangle \langle $\Psi \triangleright Q' \sim P'$ \rangle \langle *xvec #* Psi* \rangle **have** $\Psi \triangleright P' \parallel R \sim Q' \parallel T$
by(*metis bisimParPresSym bisimTransitive bisimParComm bisimE(4)*)
moreover from $suppR$ **have** $((supp(P' \parallel R))::name set) \subseteq supp((P' \parallel P''))$
by(*auto simp add: psi.supp resChainSupp*)
moreover from $suppT$ **have** $((supp(Q' \parallel T))::name set) \subseteq supp((Q' \parallel Q''))$
by(*auto simp add: psi.supp resChainSupp*)
ultimately show *?case* **by blast**
qed
ultimately show *?thesis* **by blast**
qed

lemma *structCongBisim*:
fixes $P :: ('a, 'b, 'c) psi$
and $Q :: ('a, 'b, 'c) psi$

assumes $P \equiv_s Q$

shows $P \sim Q$

using *assms*
by(*induct rule: structCong.induct*)
(*auto intro: bisimReflexive bisimSymmetric bisimTransitive bisimParComm bisimParAssoc bisimParNil bisimResNil bisimResComm bisimScopeExt bisimCasePushRes bisimInputPushRes bisimOutputPushRes bangExt*)

```

lemma bisimBangPres:
  fixes  $\Psi :: 'b$ 
    and  $P :: ('a, 'b, 'c)$  psi
    and  $Q :: ('a, 'b, 'c)$  psi

assumes  $\Psi \triangleright P \sim Q$ 
  and guarded P
  and guarded Q

shows  $\Psi \triangleright !P \sim !Q$ 
proof -
  let  $?X = \{(\Psi, R \parallel !P, R \parallel !Q) \mid \Psi P Q R. \Psi \triangleright P \sim Q \wedge \text{guarded } P \wedge \text{guarded } Q\}$ 
  let  $?Y = \{(\Psi, P, Q) \mid \Psi P P' Q' Q. \Psi \triangleright P \sim P' \wedge (\Psi, P', Q') \in ?X \wedge \Psi \triangleright Q' \sim Q\}$ 
  from assms have  $(\Psi, \mathbf{0} \parallel !P, \mathbf{0} \parallel !Q) \in ?X$  by(blast intro: bisimReflexive)

  moreover have eqvt ?X
    apply(clarsimp simp add: eqvt-def)
    apply(drule bisimClosed)
    by force
  ultimately have  $\Psi \triangleright \mathbf{0} \parallel !P \sim \mathbf{0} \parallel !Q$ 
  proof(coinduct rule: weakTransitiveCoinduct)
    case(cStatEq  $\Psi P Q$ )
    then show ?case by auto
  next
    case(cSim  $\Psi RP RQ$ )
    from  $\langle \Psi, RP, RQ \rangle \in ?X$  obtain  $P Q R$  where  $\Psi \triangleright P \sim Q$  and guarded P
and guarded Q
    and  $RP = R \parallel !P$  and  $RQ = R \parallel !Q$ 
    by auto
    note  $\langle \Psi \triangleright P \sim Q \rangle$ 
    moreover from  $\langle \text{eqvt } ?X \rangle$  have eqvt ?Y by blast
    moreover note  $\langle \text{guarded } P \rangle \langle \text{guarded } Q \rangle$  bisimE(2) bisimE(3) bisimE(4)
statEqBisim bisimClosed bisimParAssoc[THEN bisimSymmetric]
bisimParPres bisimParPresAuxSym bisimResChainPres bisimScopeExtChainSym bisimTransitive
    moreover have  $\bigwedge \Psi P Q R T. \llbracket \Psi \triangleright P \sim Q; (\Psi, Q, R) \in ?Y; \Psi \triangleright R \sim T \rrbracket$ 
 $\implies (\Psi, P, T) \in ?Y$ 
    by auto (metis bisimTransitive)
    moreover have  $\bigwedge \Psi P Q R. \llbracket \Psi \triangleright P \sim Q; \text{guarded } P; \text{guarded } Q \rrbracket \implies (\Psi, R \parallel !P, R \parallel !Q) \in ?Y$  by(blast intro: bisimReflexive)
    moreover have  $\bigwedge \Psi P \alpha P' Q. \llbracket \Psi \triangleright !P \mapsto \alpha \prec P'; \Psi \triangleright P \sim Q; \text{bn } \alpha \#* \Psi; \text{bn } \alpha \#* P; \text{bn } \alpha \#* Q; \text{guarded } Q; \text{bn } \alpha \#* \text{subject } \alpha \rrbracket \implies$ 
 $\exists Q' R T. \Psi \triangleright !Q \mapsto \alpha \prec Q' \wedge \Psi \triangleright P' \sim R \parallel !P \wedge \Psi \triangleright Q' \sim T \parallel !Q \wedge$ 
 $\Psi \triangleright R \sim T \wedge ((\text{supp } R)::\text{name set}) \subseteq$ 
supp P'  $\wedge$ 

```

```

((supp T)::name set) ⊆ supp Q'
  by(blast elim: bangDerivative)
ultimately have Ψ ▷ R || !P ~>[?Y] R || !Q
  by(rule bangPres)
with ⟨RP = R || !P⟩ ⟨RQ = R || !Q⟩ show ?case
  by blast
next
case(cExt Ψ RP RQ Ψ')
then show ?case by(blast dest: bisimE)
next
case(cSym Ψ RP RQ)
then show ?case by(blast dest: bisimE)
qed
then show ?thesis
  by(metis bisimTransitive bisimParNil bisimSymmetric bisimParComm)
qed

end

end

theory Bisim-Subst
  imports Bisim-Struct-Cong Psi-Calculi.Close-Subst
begin

  This file is a (heavily modified) variant of the theory Psi_Calculi.Bisim_Subst
  from [1].

  context env begin

  abbreviation
    bisimSubstJudge (- ▷ - ~s - [70, 70, 70] 65) where Ψ ▷ P ~s Q ≡ (Ψ, P, Q)
    ∈ closeSubst bisim
  abbreviation
    bisimSubstNilJudge (- ~s - [70, 70] 65) where P ~s Q ≡ SBottom' ▷ P ~s Q

  lemmas bisimSubstClosed[eqt] = closeSubstClosed[OF bisimEqvt]
  lemmas bisimSubstEqvt[simp] = closeSubstEqvt[OF bisimEqvt]

  lemma bisimSubstOutputPres:
    fixes Ψ :: 'b
    and P :: ('a, 'b, 'c) psi
    and Q :: ('a, 'b, 'c) psi
    and M :: 'a
    and N :: 'a

  assumes Ψ ▷ P ~s Q

  shows Ψ ▷ M⟨N⟩.P ~s M⟨N⟩.Q
    using assms closeSubstI closeSubstE bisimOutputPres by force

```

```

lemma seqSubstInputChain[simp]:
  fixes xvec :: name list
  and N    :: 'a
  and P    :: ('a, 'b, 'c) psi
  and  $\sigma$  :: (name list  $\times$  'a list) list

assumes xvec  $\#^*$   $\sigma$ 

shows seqSubs' (inputChain xvec N P)  $\sigma$  = inputChain xvec (substTerm.seqSubst
N  $\sigma$ ) (seqSubs P  $\sigma$ )
  using assms
  by(induct xvec) auto

lemma bisimSubstInputPres:
  fixes  $\Psi$   :: 'b
  and P    :: ('a, 'b, 'c) psi
  and Q    :: ('a, 'b, 'c) psi
  and M    :: 'a
  and xvec :: name list
  and N    :: 'a

assumes  $\Psi \triangleright P \sim_s Q$ 
  and xvec  $\#^*$   $\Psi$ 
  and distinct xvec

shows  $\Psi \triangleright M(\lambda^*.xvec N).P \sim_s M(\lambda^*.xvec N).Q$ 
proof(rule closeSubstI)
  fix  $\sigma$ 
  assume wellFormedSubst( $\sigma::$ (name list  $\times$  'a list) list)
  obtain p where (p  $\cdot$  xvec)  $\#^*$   $\sigma$ 
  and (p  $\cdot$  xvec)  $\#^*$  P and (p  $\cdot$  xvec)  $\#^*$  Q and (p  $\cdot$  xvec)  $\#^*$   $\Psi$  and (p  $\cdot$  xvec)  $\#^*$ 
N
  and S: set p  $\subseteq$  set xvec  $\times$  set (p  $\cdot$  xvec)
  by(rule name-list-avoiding[where c=( $\sigma$ , P, Q,  $\Psi$ , N)]) auto

from  $\langle \Psi \triangleright P \sim_s Q \rangle$  have (p  $\cdot$   $\Psi$ )  $\triangleright$  (p  $\cdot$  P)  $\sim_s$  (p  $\cdot$  Q)
  by(rule bisimSubstClosed)
with  $\langle xvec \#^* \Psi \rangle$   $\langle (p \cdot xvec) \#^* \Psi \rangle$  S have  $\Psi \triangleright (p \cdot P) \sim_s (p \cdot Q)$ 
  by simp

{
  fix Tvec :: 'a list
  from  $\langle \Psi \triangleright (p \cdot P) \sim_s (p \cdot Q) \rangle$   $\langle wellFormedSubst \sigma \rangle$  have  $\Psi \triangleright (p \cdot P)[\langle \sigma \rangle]$ 
 $\sim_s (p \cdot Q)[\langle \sigma \rangle]$ 
  by(rule closeSubstUnfold)
  moreover assume length xvec = length Tvec and distinct xvec
  ultimately have  $\Psi \triangleright ((p \cdot P)[\langle \sigma \rangle])[(p \cdot xvec)::=Tvec] \sim ((p \cdot Q)[\langle \sigma \rangle])[(p$ 
 $\cdot xvec)::=Tvec]$ 

```

```

apply –
by(drule closeSubstE[where  $\sigma = [((p \cdot xvec), Tvec)]$ ]) auto
}

with  $\langle (p \cdot xvec) \#* \sigma \rangle \langle \text{distinct } xvec \rangle$ 
have  $\Psi \triangleright (M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot P))[\langle \sigma \rangle] \sim (M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot Q))[\langle \sigma \rangle]$ 
by(force intro: bisimInputPres)
moreover from  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* P \rangle S$  have  $M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot P) = M(\lambda*xvec N).P$ 
apply(simp add: psi.inject) by(rule inputChainAlpha[symmetric]) auto
moreover from  $\langle (p \cdot xvec) \#* N \rangle \langle (p \cdot xvec) \#* Q \rangle S$  have  $M(\lambda*(p \cdot xvec) (p \cdot N)).(p \cdot Q) = M(\lambda*xvec N).Q$ 
apply(simp add: psi.inject) by(rule inputChainAlpha[symmetric]) auto
ultimately show  $\Psi \triangleright (M(\lambda*xvec N).P)[\langle \sigma \rangle] \sim (M(\lambda*xvec N).Q)[\langle \sigma \rangle]$ 
by force
qed

lemma bisimSubstCasePresAux:
fixes  $\Psi :: 'b$ 
and  $CsP :: ('c \times ('a, 'b, 'c) \text{ psi}) \text{ list}$ 
and  $CsQ :: ('c \times ('a, 'b, 'c) \text{ psi}) \text{ list}$ 

assumes  $C1: \bigwedge \varphi P. (\varphi, P) \in \text{set } CsP \implies \exists Q. (\varphi, Q) \in \text{set } CsQ \wedge \text{guarded } Q \wedge \Psi \triangleright P \sim_s Q$ 
and  $C2: \bigwedge \varphi Q. (\varphi, Q) \in \text{set } CsQ \implies \exists P. (\varphi, P) \in \text{set } CsP \wedge \text{guarded } P \wedge \Psi \triangleright P \sim_s Q$ 

shows  $\Psi \triangleright \text{Cases } CsP \sim_s \text{Cases } CsQ$ 
proof –
{
fix  $\sigma :: (\text{name list} \times 'a \text{ list}) \text{ list}$ 

assume wellFormedSubst  $\sigma$ 

have  $\Psi \triangleright \text{Cases}(\text{caseListSeqSubst } CsP \sigma) \sim \text{Cases}(\text{caseListSeqSubst } CsQ \sigma)$ 
proof(rule bisimCasePres)
fix  $\varphi P$ 
assume  $(\varphi, P) \in \text{set } (\text{caseListSeqSubst } CsP \sigma)$ 
then obtain  $\varphi' P'$  where  $(\varphi', P') \in \text{set } CsP$  and  $\varphi = \text{substCond.seqSubst } \varphi' \sigma$  and  $\text{PeqP'}: P = (P'[\langle \sigma \rangle])$ 
by(induct CsP) force+
from  $\langle (\varphi', P') \in \text{set } CsP \rangle$  obtain  $Q'$  where  $(\varphi', Q') \in \text{set } CsQ$  and guarded  $Q'$  and  $\Psi \triangleright P' \sim_s Q'$  by(blast dest: C1)
from  $\langle (\varphi', Q') \in \text{set } CsQ \rangle \langle \varphi = \text{substCond.seqSubst } \varphi' \sigma \rangle$  obtain  $Q$  where  $(\varphi, Q) \in \text{set } (\text{caseListSeqSubst } CsQ \sigma)$  and  $Q = Q'[\langle \sigma \rangle]$ 
by(induct CsQ) auto
with  $\text{PeqP'} \langle \text{guarded } Q' \rangle \langle \Psi \triangleright P' \sim_s Q' \rangle \langle \text{wellFormedSubst } \sigma \rangle$  show  $\exists Q. (\varphi, Q) \in \text{set } (\text{caseListSeqSubst } CsQ \sigma) \wedge \text{guarded } Q \wedge \Psi \triangleright P \sim Q$ 

```



```

    by(blast dest: closeSubstE guardedSeqSubst)
  next
  fix  $\varphi$   $Q$ 
  assume  $(\varphi, Q) \in \text{set } (\text{caseListSeqSubst } CsQ \sigma)$ 
  then obtain  $\varphi' Q'$  where  $(\varphi', Q') \in \text{set } CsQ$  and  $\varphi = \text{substCond.seqSubst}$ 
 $\varphi' \sigma$  and  $QeqQ': Q = Q'[\langle \sigma \rangle]$ 
    by(induct CsQ) force+
    from  $\langle (\varphi', Q') \in \text{set } CsQ \rangle$  obtain  $P'$  where  $(\varphi', P') \in \text{set } CsP$  and guarded
 $P'$  and  $\Psi \triangleright P' \sim_s Q'$  by(blast dest: C2)
    from  $\langle (\varphi', P') \in \text{set } CsP \rangle$   $\langle \varphi = \text{substCond.seqSubst } \varphi' \sigma \rangle$  obtain  $P$  where
 $(\varphi, P) \in \text{set } (\text{caseListSeqSubst } CsP \sigma)$  and  $P = P'[\langle \sigma \rangle]$ 
    by(induct CsP) auto
    with  $QeqQ' \langle \text{guarded } P' \rangle \langle \Psi \triangleright P' \sim_s Q' \rangle \langle \text{wellFormedSubst } \sigma \rangle$  show  $\exists P.$ 
 $(\varphi, P) \in \text{set } (\text{caseListSeqSubst } CsP \sigma) \wedge \text{guarded } P \wedge \Psi \triangleright P \sim Q$ 
    by(blast dest: closeSubstE guardedSeqSubst)
  qed
}
then show ?thesis
  by(intro closeSubstI) auto
qed

```

lemma *bisimSubstReflexive:*

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 

```

shows $\Psi \triangleright P \sim_s P$

```

  by(auto intro: closeSubstI bisimReflexive)

```

lemma *bisimSubstTransitive:*

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{ psi}$ 
  and  $R :: ('a, 'b, 'c) \text{ psi}$ 

```

assumes $\Psi \triangleright P \sim_s Q$

```

  and  $\Psi \triangleright Q \sim_s R$ 

```

shows $\Psi \triangleright P \sim_s R$

```

  using assms

```

```

  by(auto intro: closeSubstI closeSubstE bisimTransitive)

```

lemma *bisimSubstSymmetric:*

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{ psi}$ 

```

assumes $\Psi \triangleright P \sim_s Q$

shows $\Psi \triangleright Q \sim_s P$

```

using assms
by(auto intro: closeSubstI closeSubstE bisimE)

lemma bisimSubstCasePres:
  fixes  $\Psi :: 'b$ 
    and  $CsP :: ('c \times ('a, 'b, 'c) \text{ psi}) \text{ list}$ 
    and  $CsQ :: ('c \times ('a, 'b, 'c) \text{ psi}) \text{ list}$ 

assumes  $\text{length } CsP = \text{length } CsQ$ 
  and  $C: \bigwedge(i::\text{nat}) \varphi P \varphi' Q. \llbracket i \leq \text{length } CsP; (\varphi, P) = \text{nth } CsP \ i; (\varphi', Q) = \text{nth } CsQ \ i \rrbracket \implies \varphi = \varphi' \wedge \Psi \triangleright P \sim_s Q \wedge \text{guarded } P \wedge \text{guarded } Q$ 

shows  $\Psi \triangleright \text{Cases } CsP \sim_s \text{Cases } CsQ$ 
proof -
  {
    fix  $\varphi$ 
    and  $P$ 

    assume  $(\varphi, P) \in \text{set } CsP$ 

    with  $\langle \text{length } CsP = \text{length } CsQ \rangle$  have  $\exists Q. (\varphi, Q) \in \text{set } CsQ \wedge \text{guarded } Q \wedge \Psi \triangleright P \sim_s Q$  using  $C$ 
    proof(induct n==length CsP arbitrary: CsP CsQ rule: nat.induct)
      case zero then show ?case by simp
    next
      case(Suc n) then show ?case
      proof(cases CsP)
        case Nil then show ?thesis using  $\langle \text{Suc } n = \text{length } CsP \rangle$  by simp
      next
        case(Cons P'φ CsP')
        note  $CsPdef = \text{this}$ 
        then show ?thesis
        proof(cases CsQ)
          case Nil then show ?thesis using  $CsPdef \langle \text{length } CsP = \text{length } CsQ \rangle$ 
            by simp
        next
          case(Cons Q'φ CsQ')
          obtain  $Q' \varphi'$  where  $Q'phi: Q'φ=(\varphi',Q')$ 
            by(induct Q'φ) auto
          show ?thesis
          proof(cases P'φ = (φ,P))
            case True
            have  $0 \leq \text{length } CsP$  unfolding  $CsPdef$ 
              by simp
            moreover from  $\text{True}$  have  $(\varphi, P) = \text{nth } CsP \ 0$  using  $\langle (\varphi, P) \in \text{set } CsP \rangle$  unfolding  $CsPdef$ 
              by simp
            moreover have  $(\varphi', Q') = \text{nth } CsQ \ 0$  unfolding  $Cons \ Q'phi$  by simp
            ultimately have  $\varphi = \varphi' \wedge \Psi \triangleright P \sim_s Q' \wedge \text{guarded } P \wedge \text{guarded } Q'$ 

```

```

    by(rule Suc)
  then show ?thesis unfolding Cons Q'phi
    by(intro exI[where x=Q]) auto
next
  case False
  have n = length CsP' using ⟨Suc n = length CsP⟩ unfolding CsPdef
    by simp
  moreover have length CsP' = length CsQ' using ⟨length CsP = length
CsQ⟩ unfolding CsPdef Cons by simp
    moreover from ⟨(φ,P) ∈ set CsP⟩ False have (φ, P) ∈ set CsP'
unfolding CsPdef by simp
  moreover
  {
    fix i::nat
    and φ::'c
    and φ'::'c
    and P::('a,'b,'c) psi
    and Q::('a,'b,'c) psi
    assume i ≤ length CsP'
    and (φ, P) = CsP' ! i
    and (φ', Q) = CsQ' ! i
    then have (i+1) ≤ length CsP
    and (φ, P) = CsP ! (i+1)
    and (φ', Q) = CsQ ! (i+1)
    unfolding CsPdef Cons
    by simp+
    then have φ = φ' ∧ Ψ ▷ P ~s Q ∧ guarded P ∧ guarded Q
    by(rule Suc)
  }
  note this
  ultimately have ∃ Q. (φ, Q) ∈ set CsQ' ∧ guarded Q ∧ Ψ ▷ P ~s Q
  by(rule Suc)
  then show ?thesis unfolding Cons by auto
qed
qed
qed
qed
}
note this
moreover
{
  fix φ
  and Q

  assume (φ, Q) ∈ set CsQ

  with ⟨length CsP = length CsQ⟩ have ∃ P. (φ, P) ∈ set CsP ∧ guarded P ∧
Ψ ▷ P ~s Q using C
  proof(induct n==length CsQ arbitrary: CsP CsQ rule: nat.induct)

```

```

case zero then show ?case by simp
next
case(Suc n) then show ?case
proof(cases CsQ)
  case Nil then show ?thesis using ⟨Suc n = length CsQ⟩ by simp
next
  case(Cons Q'φ CsQ')
  note CsPdef = this
  then show ?thesis
  proof(cases CsP)
    case Nil then show ?thesis using CsPdef ⟨length CsP = length CsQ⟩
      by simp
    next
    case(Cons P'φ CsP')
    obtain P' φ' where P'phi: P'φ=(φ',P')
      by(induct P'φ) auto
    show ?thesis
    proof(cases Q'φ = (φ,Q))
      case True
      have 0 ≤ length CsP unfolding CsPdef
        by simp
      moreover have (φ', P') = nth CsP 0 unfolding Cons P'phi by simp
      moreover from True have (φ, Q) = nth CsQ 0 using ⟨(φ, Q) ∈ set
CsQ⟩ unfolding CsPdef
        by simp
      ultimately have φ' = φ ∧ Ψ ▷ P' ~s Q ∧ guarded P' ∧ guarded Q
        by(rule Suc)
      then show ?thesis unfolding Cons P'phi
        by(intro exI[where x=P']) auto
      next
      case False
      have n = length CsQ' using ⟨Suc n = length CsQ⟩ unfolding CsPdef
        by simp
      moreover have length CsP' = length CsQ' using ⟨length CsP = length
CsQ⟩ unfolding CsPdef Cons by simp
      moreover from ⟨(φ,Q) ∈ set CsQ⟩ False have (φ, Q) ∈ set CsQ'
unfolding CsPdef by simp
      moreover
      {
        fix i::nat
        and φ::'c
        and φ'::'c
        and P::('a,'b,'c) psi
        and Q::('a,'b,'c) psi
        assume i ≤ length CsP'
        and (φ, P) = CsP' ! i
        and (φ', Q) = CsQ' ! i
        then have (i+1) ≤ length CsP
          and (φ, P) = CsP ! (i+1)

```

```

    and  $(\varphi', Q) = CsQ ! (i+1)$ 
    unfolding CsPdef Cons
    by simp+
    then have  $\varphi = \varphi' \wedge \Psi \triangleright P \sim_s Q \wedge \text{guarded } P \wedge \text{guarded } Q$ 
    by(rule Suc)
  }
  note this
  ultimately have  $\exists P. (\varphi, P) \in \text{set } CsP' \wedge \text{guarded } P \wedge \Psi \triangleright P \sim_s Q$ 
  by(rule Suc)
  then show ?thesis unfolding Cons by auto
qed
qed
qed
qed
}
ultimately show ?thesis
by(rule bisimSubstCasePresAux)
qed

```

lemma *bisimSubstParPres:*

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{ psi}$ 
  and  $R :: ('a, 'b, 'c) \text{ psi}$ 

```

assumes $\Psi \triangleright P \sim_s Q$

shows $\Psi \triangleright P \parallel R \sim_s Q \parallel R$
 using *assms closeSubstI closeSubstE bisimParPres* by *force*

lemma *bisimSubstResPres:*

```

  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) \text{ psi}$ 
  and  $Q :: ('a, 'b, 'c) \text{ psi}$ 
  and  $x :: \text{name}$ 

```

assumes $\Psi \triangleright P \sim_s Q$
 and $x \# \Psi$

shows $\Psi \triangleright (\nu x)P \sim_s (\nu x)Q$

proof(*rule closeSubstI*)

fix $\sigma :: (\text{name list} \times 'a \text{ list}) \text{ list}$

assume *wellFormedSubst* σ

obtain $y :: \text{name}$ **where** $y \# \Psi$ **and** $y \# P$ **and** $y \# Q$ **and** $y \# \sigma$
 by(*generate-fresh name*) (*auto simp add: fresh-prod*)

from $\langle \Psi \triangleright P \sim_s Q \rangle$ **have** $([(x, y)] \cdot \Psi) \triangleright ([(x, y)] \cdot P) \sim_s ([(x, y)] \cdot Q)$
 by(*rule bisimSubstClosed*)

```

with  $\langle x \# \Psi \rangle \langle y \# \Psi \rangle$  have  $\Psi \triangleright ((x, y) \cdot P) \sim_s ((x, y) \cdot Q)$ 
  by simp
then have  $\Psi \triangleright ((x, y) \cdot P)[\langle \sigma \rangle] \sim ((x, y) \cdot Q)[\langle \sigma \rangle]$  using  $\langle wellFormedSubst \sigma \rangle$ 
  by(rule closeSubstE)
then have  $\Psi \triangleright (\nu y)((x, y) \cdot P)[\langle \sigma \rangle] \sim (\nu y)((x, y) \cdot Q)[\langle \sigma \rangle]$  using  $\langle y \# \Psi \rangle$ 
  by(rule bisimResPres)
with  $\langle y \# P \rangle \langle y \# Q \rangle \langle y \# \sigma \rangle$ 
show  $\Psi \triangleright ((\nu x)P)[\langle \sigma \rangle] \sim ((\nu x)Q)[\langle \sigma \rangle]$ 
  by(simp add: alphaRes)
qed

```

lemma *bisimSubstBangPres*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 

```

```

assumes  $\Psi \triangleright P \sim_s Q$ 
  and guarded P
  and guarded Q

```

```

shows  $\Psi \triangleright !P \sim_s !Q$ 
  using assms closeSubstI closeSubstE bisimBangPres guardedSeqSubst by force

```

lemma *substNil[*simp*]*:

```

fixes xvec :: name list
  and Tvec :: 'a list

```

```

assumes wellFormedSubst  $\sigma$ 
  and distinct xvec

```

```

shows  $(\mathbf{0}[\langle \sigma \rangle]) = \mathbf{0}$ 
  using assms
  by simp

```

lemma *bisimSubstParNil*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 

```

```

shows  $\Psi \triangleright P \parallel \mathbf{0} \sim_s P$ 
  by(auto intro!: closeSubstI bisimParNil)

```

lemma *bisimSubstParComm*:

```

fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 

```

```

shows  $\Psi \triangleright P \parallel Q \sim_s Q \parallel P$ 

```

```

by(auto intro!: closeSubstI bisimParComm)

lemma bisimSubstParAssoc:
  fixes  $\Psi :: 'b$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 
  and  $R :: ('a, 'b, 'c) psi$ 

shows  $\Psi \triangleright (P \parallel Q) \parallel R \sim_s P \parallel (Q \parallel R)$ 
by(auto intro!: closeSubstI bisimParAssoc)

lemma bisimSubstResNil:
  fixes  $\Psi :: 'b$ 
  and  $x :: name$ 

shows  $\Psi \triangleright (\nu x)\mathbf{0} \sim_s \mathbf{0}$ 
proof(rule closeSubstI)
  fix  $\sigma :: (name list \times 'a list) list$ 

  assume wellFormedSubst  $\sigma$ 
  obtain  $y::name$  where  $y \# \Psi$  and  $y \# \sigma$ 
  by(generate-fresh name) (auto simp add: fresh-prod)
  have  $\Psi \triangleright (\nu y)\mathbf{0} \sim \mathbf{0}$  by(rule bisimResNil)
  with  $\langle y \# \sigma \rangle \langle wellFormedSubst \sigma \rangle$  show  $\Psi \triangleright ((\nu x)\mathbf{0})[\langle \sigma \rangle] \sim \mathbf{0}[\langle \sigma \rangle]$ 
  by(subst alphaRes[of y]) auto
qed

lemma seqSubst2:
  fixes  $x :: name$ 
  and  $P :: ('a, 'b, 'c) psi$ 

assumes wellFormedSubst  $\sigma$ 
  and  $x \# \sigma$ 
  and  $x \# P$ 

shows  $x \# P[\langle \sigma \rangle]$ 
using assms
by(induct  $\sigma$  arbitrary:  $P$ , auto) (blast dest: subst2)

notation substTerm.seqSubst (-[\langle - \rangle] [100, 100] 100)

lemma bisimSubstScopeExt:
  fixes  $\Psi :: 'b$ 
  and  $x :: name$ 
  and  $P :: ('a, 'b, 'c) psi$ 
  and  $Q :: ('a, 'b, 'c) psi$ 

assumes  $x \# P$ 

```

```

shows  $\Psi \triangleright (\nu x)(P \parallel Q) \sim_s P \parallel (\nu x)Q$ 
proof(rule closeSubstI)
  fix  $\sigma :: (\text{name list} \times \text{'a list}) \text{ list}$ 

  assume wellFormedSubst  $\sigma$ 
  obtain  $y :: \text{name}$  where  $y \# \Psi$  and  $y \# \sigma$  and  $y \# P$  and  $y \# Q$ 
    by(generate-fresh name) (auto simp add: fresh-prod)
  moreover from  $\langle \text{wellFormedSubst } \sigma \rangle \langle y \# \sigma \rangle \langle y \# P \rangle$  have  $y \# P[\langle \sigma \rangle]$ 
    by(rule seqSubst2)
  then have  $\Psi \triangleright (\nu y)((P[\langle \sigma \rangle]) \parallel (((x, y) \cdot Q)[\langle \sigma \rangle])) \sim (P[\langle \sigma \rangle]) \parallel (\nu y)((x, y) \cdot Q)[\langle \sigma \rangle]$ 
    by(rule bisimScopeExt)
  with  $\langle x \# P \rangle \langle y \# P \rangle \langle y \# Q \rangle \langle y \# \sigma \rangle$  show  $\Psi \triangleright ((\nu x)(P \parallel Q))[\langle \sigma \rangle] \sim (P \parallel (\nu x)Q)[\langle \sigma \rangle]$ 
    apply(subst alphaRes[of y], simp)
    apply(subst alphaRes[of y Q], simp)
    by(simp add: eqvts)
qed

```

```

lemma bisimSubstCasePushRes:
  fixes  $x :: \text{name}$ 
    and  $\Psi :: \text{'b}$ 
    and  $Cs :: (\text{'c} \times (\text{'a}, \text{'b}, \text{'c}) \text{ psi}) \text{ list}$ 

```

```

assumes  $x \# \text{map fst } Cs$ 

```

```

shows  $\Psi \triangleright (\nu x)(\text{Cases } Cs) \sim_s \text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs$ 
proof(rule closeSubstI)
  fix  $\sigma :: (\text{name list} \times \text{'a list}) \text{ list}$ 

```

```

  assume wellFormedSubst  $\sigma$ 
  obtain  $y :: \text{name}$  where  $y \# \Psi$  and  $y \# \sigma$  and  $y \# Cs$ 
    by(generate-fresh name) (auto simp add: fresh-prod)

```

```

  {
    fix  $x :: \text{name}$ 
      and  $Cs :: (\text{'c} \times (\text{'a}, \text{'b}, \text{'c}) \text{ psi}) \text{ list}$ 
      and  $\sigma :: (\text{name list} \times \text{'a list}) \text{ list}$ 

```

```

    assume  $x \# \sigma$ 

```

```

    then have  $(\text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs)[\langle \sigma \rangle] = \text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) (\text{caseListSeqSubst } Cs \sigma)$ 
      by(induct Cs) auto

```

```

  }
  note  $C1 = \text{this}$ 

```

```

  {
    fix  $x :: \text{name}$ 

```



```

and  $y$  :: name
and  $Cs$  :: ('c × ('a, 'b, 'c) psi) list

assume  $x \# \text{map fst } Cs$ 
and  $y \# \text{map fst } Cs$ 
and  $y \# Cs$ 

then have ( $\text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs = \text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu y)P)) ([x, y] \cdot Cs)$ )
by( $\text{induct } Cs$ ) ( $\text{auto simp add: fresh-list-cons alphaRes}$ )
}
note  $C2 = \text{this}$ 

from  $\langle y \# Cs \rangle$  have  $y \# \text{map fst } Cs$  by( $\text{induct } Cs$ ) ( $\text{auto simp add: fresh-list-cons fresh-list-nil}$ )
from  $\langle y \# Cs \rangle \langle y \# \sigma \rangle \langle x \# \text{map fst } Cs \rangle \langle \text{wellFormedSubst } \sigma \rangle$  have  $y \# \text{map fst } (\text{caseListSeqSubst } ([x, y] \cdot Cs) \sigma)$ 
by( $\text{induct } Cs$ ) ( $\text{auto intro: substCond.seqSubst2 simp add: fresh-list-cons fresh-list-nil fresh-prod}$ )
then have  $\Psi \triangleright (\nu y)(\text{Cases}(\text{caseListSeqSubst } ([x, y] \cdot Cs) \sigma)) \sim \text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu y)P)) (\text{caseListSeqSubst } ([x, y] \cdot Cs) \sigma)$ 
by( $\text{rule bisimCasePushRes}$ )

with  $\langle y \# Cs \rangle \langle x \# \text{map fst } Cs \rangle \langle y \# \text{map fst } Cs \rangle \langle y \# \sigma \rangle \langle \text{wellFormedSubst } \sigma \rangle$ 
show  $\Psi \triangleright ((\nu x)(\text{Cases } Cs))[\langle \sigma \rangle] \sim (\text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs)[\langle \sigma \rangle]$ 
apply( $\text{subst } C2[\text{of } x \text{ } Cs \text{ } y]$ )
apply  $\text{assumption+}$ 
apply( $\text{subst } C1$ )
apply  $\text{assumption+}$ 
apply( $\text{subst alphaRes}[\text{of } y], \text{simp}$ )
by( $\text{simp add: eqvts}$ )
qed

lemma  $\text{bisimSubstOutputPushRes}$ :
fixes  $x$  :: name
and  $\Psi$  :: 'b
and  $M$  :: 'a
and  $N$  :: 'a
and  $P$  :: ('a, 'b, 'c) psi

assumes  $x \# M$ 
and  $x \# N$ 

shows  $\Psi \triangleright (\nu x)(M\langle N \rangle.P) \sim_s M\langle N \rangle.(\nu x)P$ 
proof( $\text{rule closeSubstI}$ )
fix  $\sigma$ :: (name list × 'a list) list

assume  $\text{wellFormedSubst } \sigma$ 
obtain  $y$ ::name where  $y \# \Psi$  and  $y \# \sigma$  and  $y \# P$  and  $y \# M$  and  $y \# N$ 

```

by(*generate-fresh name*) (*auto simp add: fresh-prod*)
from $\langle \text{wellFormedSubst } \sigma \rangle \langle y \# M \rangle \langle y \# \sigma \rangle$ **have** $y \# M[\langle \sigma \rangle]$ **by** *auto*
moreover from $\langle \text{wellFormedSubst } \sigma \rangle \langle y \# N \rangle \langle y \# \sigma \rangle$ **have** $y \# N[\langle \sigma \rangle]$ **by** *auto*
ultimately have $\Psi \triangleright (\nu y)((M[\langle \sigma \rangle])(N[\langle \sigma \rangle])).(((x, y) \cdot P)[\langle \sigma \rangle]) \sim$
 $(M[\langle \sigma \rangle])(N[\langle \sigma \rangle]).((\nu y)(([x, y] \cdot P)[\langle \sigma \rangle]))$
by(*rule bisimOutputPushRes*)
with $\langle y \# M \rangle \langle y \# N \rangle \langle y \# P \rangle \langle x \# M \rangle \langle x \# N \rangle \langle y \# \sigma \rangle \langle \text{wellFormedSubst } \sigma \rangle$
show $\Psi \triangleright ((\nu x)(M\langle N \rangle.P))[\langle \sigma \rangle] \sim (M\langle N \rangle.(\nu x)P)[\langle \sigma \rangle]$
apply(*subst alphaRes[of y], simp*)
apply(*subst alphaRes[of y P], simp*)
by(*simp add: eqvts*)
qed

lemma *bisimSubstInputPushRes:*

fixes $x \quad :: \text{ name}$
and $\Psi \quad :: 'b$
and $M \quad :: 'a$
and $xvec \quad :: \text{ name list}$
and $N \quad :: 'a$

assumes $x \# M$
and $x \# xvec$
and $x \# N$

shows $\Psi \triangleright ((\nu x)(M(\lambda * xvec N).P)) \sim_s M(\lambda * xvec N).(\nu x)P$

proof(*rule closeSubstI*)

fix $\sigma :: (\text{ name list } \times 'a \text{ list}) \text{ list}$

assume *wellFormedSubst* σ

obtain $y :: \text{ name}$ **where** $y \# \Psi$ **and** $y \# \sigma$ **and** $y \# P$ **and** $y \# M$ **and** $y \# xvec$
and $y \# N$

by(*generate-fresh name*) (*auto simp add: fresh-prod*)

obtain $p :: \text{ name prm}$ **where** $(p \cdot xvec) \# * N$ **and** $(p \cdot xvec) \# * P$ **and** $x \# (p \cdot xvec)$ **and** $y \# (p \cdot xvec)$ **and** $(p \cdot xvec) \# * \sigma$

and $S: \text{ set } p \subseteq \text{ set } xvec \times \text{ set}(p \cdot xvec)$

by(*rule name-list-avoiding[where c=(N, P, x, y, sigma)]*) *auto*

from $\langle \text{wellFormedSubst } \sigma \rangle \langle y \# M \rangle \langle y \# \sigma \rangle$ **have** $y \# M[\langle \sigma \rangle]$ **by** *auto*

moreover note $\langle y \# (p \cdot xvec) \rangle$

moreover from $\langle y \# N \rangle$ **have** $(p \cdot y) \# (p \cdot N)$ **by**(*simp add: pt-fresh-bij[OF pt-name-inst, OF at-name-inst]*)

with $\langle y \# xvec \rangle \langle y \# (p \cdot xvec) \rangle S$ **have** $y \# p \cdot N$ **by** *simp*

with $\langle \text{wellFormedSubst } \sigma \rangle$ **have** $y \# (p \cdot N)[\langle \sigma \rangle]$ **using** $\langle y \# \sigma \rangle$ **by** *auto*

ultimately have $\Psi \triangleright (\nu y)((M[\langle \sigma \rangle])(\lambda *(p \cdot xvec) ((p \cdot N)[\langle \sigma \rangle])).(((x, y) \cdot (p \cdot P))[\langle \sigma \rangle])) \sim (M[\langle \sigma \rangle])(\lambda *(p \cdot xvec) ((p \cdot N)[\langle \sigma \rangle])).((\nu y)(([x, y] \cdot p \cdot P)[\langle \sigma \rangle]))$

by(*rule bisimInputPushRes*)

with $\langle y \# M \rangle \langle y \# N \rangle \langle y \# P \rangle \langle x \# M \rangle \langle x \# N \rangle \langle y \# xvec \rangle \langle x \# xvec \rangle \langle (p \cdot xvec) \rangle$

```

#* N > ⟨p · xvec⟩ #* P >
  ⟨x # (p · xvec)⟩ ⟨y # (p · xvec)⟩ ⟨y # σ⟩ ⟨p · xvec⟩ #* σ > S ⟨wellFormedSubst σ⟩
show Ψ ▷ ((νx)(M(λ*xvec N).P))[<σ>] ~ (M(λ*xvec N).(νx)P)[<σ>]
  apply(subst inputChainAlpha')
  apply assumption+
  apply(subst inputChainAlpha'[of p xvec])
  apply(simp add: abs-fresh-star)
  apply assumption+
  apply(simp add: eqvts)
  apply(subst alphaRes[of y], simp)
  apply(simp add: inputChainFresh)
  apply(simp add: freshChainSimps)
  apply(subst alphaRes[of y (p · P)])
  apply(simp add: freshChainSimps)
  by(simp add: freshChainSimps eqvts)
qed

```

lemma bisimSubstResComm:

```

fixes x :: name
and y :: name

```

```

shows Ψ ▷ (νx)((νy)P) ~s (νy)((νx)P)
proof(cases x = y)
  assume x = y
  then show ?thesis by(force intro: bisimSubstReflexive)
next
  assume x ≠ y
  show ?thesis
  proof(rule closeSubstI)
    fix σ:: (name list × 'a list) list
    assume wellFormedSubst σ

```

```

    obtain x'::name where x' # Ψ and x' # σ and x' # P and x ≠ x' and y ≠ x'
      by(generate-fresh name) (auto simp add: fresh-prod)
    obtain y'::name where y' # Ψ and y' # σ and y' # P and x ≠ y' and y ≠
y' and x' ≠ y'
      by(generate-fresh name) (auto simp add: fresh-prod)

```

```

    have Ψ ▷ (νx')((νy')(((x, x') · [(y, y')] · P)[<σ>])) ~ (νy')((νx')(((x, x')
· [(y, y')] · P)[<σ>]))
      by(rule bisimResComm)
    moreover from x' # P > y' # P > x ≠ y' > x' ≠ y' have (νx)((νy)P) =
(νx')((νy')(((x, x') · [(y, y')] · P)))
      apply(subst alphaRes[of y' P], simp)
      by(subst alphaRes[of x']) (auto simp add: abs-fresh fresh-left calc-atm eqvts)
    moreover from x' # P > y' # P > y ≠ x' > x ≠ y' > x' ≠ y' > x ≠ x' > x ≠ y
have (νy)((νx)P) = (νy')((νx')(((x, x') · [(y, y')] · P)))
      apply(subst alphaRes[of x' P], simp)

```

```

apply(subst alphaRes[of y'], simp add: abs-fresh fresh-left calc-atm)
apply(simp add: eqvts calc-atm)
by(subst perm-compose) (simp add: calc-atm)

ultimately show  $\Psi \triangleright ((\nu x)((\nu y)P))[\langle \sigma \rangle] \sim ((\nu y)((\nu x)P))[\langle \sigma \rangle]$ 
using  $\langle \text{wellFormedSubst } \sigma \rangle \langle x' \# \sigma \rangle \langle y' \# \sigma \rangle$ 
by simp
qed
qed

lemma bisimSubstExtBang:
fixes  $\Psi :: 'b$ 
and  $P :: ('a, 'b, 'c) \text{ psi}$ 

assumes guarded P

shows  $\Psi \triangleright !P \sim_s P \parallel !P$ 
using assms closeSubstI bangExt guardedSeqSubst by force

lemma structCongBisimSubst:
fixes  $P :: ('a, 'b, 'c) \text{ psi}$ 
and  $Q :: ('a, 'b, 'c) \text{ psi}$ 

assumes  $P \equiv_s Q$ 

shows  $P \sim_s Q$ 
using assms
by(induct rule: structCong.induct)
(auto intro: bisimSubstReflexive bisimSubstSymmetric bisimSubstTransitive bisim-
SubstParComm bisimSubstParAssoc bisimSubstParNil bisimSubstResNil bisimSub-
stResComm bisimSubstScopeExt bisimSubstCasePushRes bisimSubstInputPushRes
bisimSubstOutputPushRes bisimSubstExtBang)

end

end

theory Broadcast-Thms
imports Broadcast-Chain Broadcast-Frame Semantics Simulation Bisimulation
Sim-Pres
Sim-Struct-Cong Bisim-Pres Bisim-Struct-Cong Bisim-Subst
begin

context env
begin

```

2.1 Syntax

2.1.1 Psi calculus agents – Definition 2

M, N range over message terms. P, Q range over processes. C ranges over cases.

- Output: $M\langle N\rangle.P$
- Input: $M(\lambda*xvec N).P$
- Case: $Case\ C$
- Par: $P \parallel Q$
- Res: $(\nu x)P$
- Assert: $\{\Psi\}$
- Bang: $!P$

- Cases: $\square\ \varphi \Rightarrow P\ C$

2.1.2 Parameters – Definition 1

- Channel equivalence: $M \leftrightarrow N$
- Composition: $\Psi_P \otimes \Psi_Q$
- Unit: 1
- Entailment: $\Psi \vdash \varphi$

2.1.3 Extra predicates for broadcast – Definition 5

- Output connectivity: $M \preceq N$
- Input connectivity: $M \succeq N$

2.1.4 Transitions – Definition 3

- $\Psi \triangleright P \mapsto \alpha\ P'$

2.1.5 Actions (α) – Definition 7

- Input: $M(N)$
- Output: $M(\nu * xvec)\langle N \rangle$
- Broadcast input: $!M(N)$
- Broadcast output: $!M(\nu * xvec)\langle N \rangle$
- Silent action: τ

2.2 Semantics

2.2.1 Basic Psi semantics – Table 1

- Theorem *Input*:

$$\begin{aligned} & \llbracket \Psi \vdash M \leftrightarrow K; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N; \text{length } xvec = \text{length } Tvec \rrbracket \\ & \implies \Psi \triangleright M(\lambda * xvec N).P \mapsto K(N[xvec ::= Tvec]) \prec P[xvec ::= Tvec] \end{aligned}$$

- Theorem *Output*:

$$\Psi \vdash M \leftrightarrow K \implies \Psi \triangleright M\langle N \rangle.P \mapsto K\langle N \rangle \prec P$$

- Theorem *Case*:

$$\llbracket \Psi \triangleright P \mapsto Rs; (\varphi, P) \text{ mem } Cs; \Psi \vdash \varphi; \text{guarded } P \rrbracket \implies \Psi \triangleright \text{Cases } Cs \mapsto Rs$$

- Theorems *Par1* and *Par2*:

$$\begin{aligned} & \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \alpha \prec P'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \text{bn } \alpha \# * Q; A_Q \# * \\ & \Psi; \\ & A_Q \# * P; A_Q \# * \alpha \rrbracket \\ & \implies \Psi \triangleright P \parallel Q \mapsto \alpha \prec P' \parallel Q \end{aligned}$$

$$\begin{aligned} & \llbracket \Psi \otimes \Psi_P \triangleright Q \mapsto \alpha \prec Q'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \text{bn } \alpha \# * P; A_P \# * \\ & \Psi; \\ & A_P \# * Q; A_P \# * \alpha \rrbracket \\ & \implies \Psi \triangleright P \parallel Q \mapsto \alpha \prec P \parallel Q' \end{aligned}$$

- Theorems *Comm1* and *Comm2*:

$$\begin{aligned} & \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(N) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \\ & \Psi \otimes \Psi_P \triangleright Q \mapsto K(\nu * xvec)\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \\ & \Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K; A_P \# * \Psi; A_P \# * P; A_P \# * Q; A_P \# * M; A_P \# * \\ & A_Q; A_Q \# * \Psi; \\ & A_Q \# * P; A_Q \# * Q; A_Q \# * K; xvec \# * P \rrbracket \\ & \implies \Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu * xvec)P' \parallel Q' \end{aligned}$$

$$\begin{aligned}
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto M(\nu^*xvec)\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto K\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; \Psi \otimes \Psi_P \otimes \\
& \Psi_Q \vdash M \leftrightarrow K; \\
& A_P \#^* \Psi; A_P \#^* P; A_P \#^* Q; A_P \#^* M; A_P \#^* A_Q; A_Q \#^* \Psi; A_Q \#^* P; A_Q \\
& \#^* Q; \\
& A_Q \#^* K; xvec \#^* Q \\
& \implies \Psi \triangleright P \parallel Q \mapsto \tau \prec (\nu^*xvec)P' \parallel Q'
\end{aligned}$$

- Theorem *Open*:

$$\begin{aligned}
& \llbracket \Psi \triangleright P \mapsto M(\nu^*(xvec \ @ \ yvec))\langle N \rangle \prec P'; x \in \text{supp } N; x \# \Psi; x \# M; x \# \\
& xvec; \\
& x \# yvec \\
& \implies \Psi \triangleright (\nu x)P \mapsto M(\nu^*(xvec \ @ \ x \ # \ yvec))\langle N \rangle \prec P'
\end{aligned}$$

- Theorem *Scope*:

$$\llbracket \Psi \triangleright P \mapsto \alpha \prec P'; x \# \Psi; x \# \alpha \rrbracket \implies \Psi \triangleright (\nu x)P \mapsto \alpha \prec (\nu x)P'$$

- Theorem *Bang*:

$$\llbracket \Psi \triangleright P \parallel !P \mapsto Rs; \text{guarded } P \rrbracket \implies \Psi \triangleright !P \mapsto Rs$$

2.2.2 Broadcast rules – Table 2

- Theorem *BrInput*:

$$\begin{aligned}
& \llbracket \Psi \vdash K \succeq M; \text{distinct } xvec; \text{set } xvec \subseteq \text{supp } N; \text{length } xvec = \text{length } Tvec \\
& \implies \Psi \triangleright M(\lambda^*xvec N).P \mapsto \imath K\langle N[xvec ::= Tvec] \rangle \prec P[xvec ::= Tvec]
\end{aligned}$$

- Theorem *BrOutput*:

$$\Psi \vdash M \preceq K \implies \Psi \triangleright M\langle N \rangle.P \mapsto \imath K\langle N \rangle \prec P$$

- Theorem *BrMerge*:

$$\begin{aligned}
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \imath M\langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto \imath M\langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; A_P \#^* \Psi; \\
& A_P \#^* P; \\
& A_P \#^* Q; A_P \#^* M; A_P \#^* A_Q; A_Q \#^* \Psi; A_Q \#^* P; A_Q \#^* Q; A_Q \#^* M \\
& \implies \Psi \triangleright P \parallel Q \mapsto \imath M\langle N \rangle \prec P' \parallel Q'
\end{aligned}$$

- Theorems *BrComm1* and *BrComm2*:

$$\begin{aligned}
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\langle N \rangle) \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\langle \nu * xvec \rangle) \langle N \rangle \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; A_P \# * \Psi; \\
& A_P \# * P; A_P \# * Q; A_P \# * M; A_P \# * A_Q; A_Q \# * \Psi; A_Q \# * P; A_Q \# * Q; A_Q \\
& \# * M; \\
& xvec \# * P \rrbracket \\
\implies & \Psi \triangleright P \parallel Q \mapsto \imath M(\langle \nu * xvec \rangle) \langle N \rangle \prec P' \parallel Q'
\end{aligned}$$

$$\begin{aligned}
& \llbracket \Psi \otimes \Psi_Q \triangleright P \mapsto \imath M(\langle \nu * xvec \rangle) \langle N \rangle \prec P'; \text{extractFrame } P = \langle A_P, \Psi_P \rangle; \\
& \Psi \otimes \Psi_P \triangleright Q \mapsto \imath M(\langle N \rangle) \prec Q'; \text{extractFrame } Q = \langle A_Q, \Psi_Q \rangle; A_P \# * \Psi; \\
& A_P \# * P; \\
& A_P \# * Q; A_P \# * M; A_P \# * A_Q; A_Q \# * \Psi; A_Q \# * P; A_Q \# * Q; A_Q \# * M; \\
& xvec \# * Q \rrbracket \\
\implies & \Psi \triangleright P \parallel Q \mapsto \imath M(\langle \nu * xvec \rangle) \langle N \rangle \prec P' \parallel Q'
\end{aligned}$$

- Theorem *BrClose*:

$$\begin{aligned}
& \llbracket \Psi \triangleright P \mapsto \imath M(\langle \nu * xvec \rangle) \langle N \rangle \prec P'; x \in \text{supp } M; x \# \Psi \rrbracket \\
\implies & \Psi \triangleright (\nu x)P \mapsto \tau \prec (\nu x)(\langle \nu * xvec \rangle P')
\end{aligned}$$

- Theorem *BrOpen*:

$$\begin{aligned}
& \llbracket \Psi \triangleright P \mapsto \imath M(\langle \nu * (xvec @ yvec) \rangle) \langle N \rangle \prec P'; x \in \text{supp } N; x \# \Psi; x \# M; x \# \\
& xvec; \\
& x \# yvec \rrbracket \\
\implies & \Psi \triangleright (\nu x)P \mapsto \imath M(\langle \nu * (xvec @ x \# yvec) \rangle) \langle N \rangle \prec P'
\end{aligned}$$

2.2.3 Requirements for broadcast – Definition 6

- Theorem *chanOutConSupp*:

$$\Psi \vdash M \preceq N \implies \text{supp } N \subseteq \text{supp } M$$

- Theorem *chanInConSupp*:

$$\Psi \vdash N \succeq M \implies \text{supp } N \subseteq \text{supp } M$$

2.2.4 Strong bisimulation – Definition 4

- Theorem *bisim.step*:

$$\begin{aligned}
& \llbracket \text{insertAssertion } (\text{extractFrame } P) \Psi \simeq_F \text{insertAssertion } (\text{extractFrame } Q) \\
& \Psi; \\
& \Psi \triangleright P \rightsquigarrow [\text{bisim}] Q; \forall \Psi'. \Psi \otimes \Psi' \triangleright P \sim Q; \Psi \triangleright Q \sim P \rrbracket \\
\implies & \Psi \triangleright P \sim Q
\end{aligned}$$

2.3 Theorems

2.3.1 Congruence properties of strong bisimulation – Theorem 8

- Theorem *bisimOutputPres*:

$$\Psi \triangleright P \sim Q \implies \Psi \triangleright M\langle N \rangle.P \sim M\langle N \rangle.Q$$

- Theorem *bisimInputPres*:

$$\begin{aligned} & (\bigwedge Tvec. \text{length } xvec = \text{length } Tvec \implies \Psi \triangleright P[xvec ::= Tvec] \sim Q[xvec ::= Tvec]) \\ & \implies \\ & \Psi \triangleright M(\lambda * xvec N).P \sim M(\lambda * xvec N).Q \end{aligned}$$

- Theorem *bisimCasePres*:

$$\begin{aligned} & \llbracket \bigwedge \varphi P. (\varphi, P) \text{ mem } CsP \implies \exists Q. (\varphi, Q) \text{ mem } CsQ \wedge \text{guarded } Q \wedge \Psi \triangleright P \\ & \sim Q; \\ & \bigwedge \varphi Q. (\varphi, Q) \text{ mem } CsQ \implies \exists P. (\varphi, P) \text{ mem } CsP \wedge \text{guarded } P \wedge \Psi \triangleright P \\ & \sim Q \rrbracket \\ & \implies \Psi \triangleright \text{Cases } CsP \sim \text{Cases } CsQ \end{aligned}$$

- Theorems *bisimParPres* and *bisimParPresSym*:

$$\Psi \triangleright P \sim Q \implies \Psi \triangleright P \parallel R \sim Q \parallel R$$

$$\Psi \triangleright P \sim Q \implies \Psi \triangleright R \parallel P \sim R \parallel Q$$

- Theorem *bisimResPres*:

$$\llbracket \Psi \triangleright P \sim Q; x \# \Psi \rrbracket \implies \Psi \triangleright (\nu x)P \sim (\nu x)Q$$

- Theorem *bisimBangPres*:

$$\llbracket \Psi \triangleright P \sim Q; \text{guarded } P; \text{guarded } Q \rrbracket \implies \Psi \triangleright !P \sim !Q$$

2.3.2 Strong congruence, bisimulation closed under substitution – Definition 9

- Theorem *closeSubst_def*:

$$\begin{aligned} & \text{closeSubst Rel} \equiv \\ & \{(\Psi, P, Q) \mid \Psi P Q. \forall \sigma. \text{wellFormedSubst } \sigma \longrightarrow (\Psi, P[\langle \sigma \rangle], Q[\langle \sigma \rangle]) \in \\ & \text{Rel}\} \end{aligned}$$

- $\Psi \triangleright P \sim_s Q$

2.3.3 Strong congruence is a process congruence for all Ψ – Theorem 10

- Theorem *bisimSubstOutputPres*:

$$\Psi \triangleright P \sim_s Q \implies \Psi \triangleright M\langle N \rangle.P \sim_s M\langle N \rangle.Q$$

- Theorem *bisimSubstInputPres*:

$$\llbracket \Psi \triangleright P \sim_s Q; \text{vec } \sharp^* \Psi; \text{distinct vec} \rrbracket \implies \Psi \triangleright M(\lambda^* \text{vec } N).P \sim_s M(\lambda^* \text{vec } N).Q$$

- Theorem *bisimSubstCasePres*:

$$\begin{aligned} & \llbracket \text{length } CsP = \text{length } CsQ; \\ & \quad \bigwedge i \varphi P \varphi' Q. \\ & \quad \llbracket i \leq \text{length } CsP; (\varphi, P) = CsP ! i; (\varphi', Q) = CsQ ! i \rrbracket \\ & \quad \implies \varphi = \varphi' \wedge \Psi \triangleright P \sim_s Q \wedge \text{guarded } P \wedge \text{guarded } Q \rrbracket \\ & \implies \Psi \triangleright \text{Cases } CsP \sim_s \text{Cases } CsQ \end{aligned}$$

- Theorem *bisimSubstParPres*:

$$\Psi \triangleright P \sim_s Q \implies \Psi \triangleright P \parallel R \sim_s Q \parallel R$$

- Theorem *bisimSubstResPres*:

$$\llbracket \Psi \triangleright P \sim_s Q; x \sharp \Psi \rrbracket \implies \Psi \triangleright (\nu x)P \sim_s (\nu x)Q$$

- Theorem *bisimSubstBangPres*:

$$\llbracket \Psi \triangleright P \sim_s Q; \text{guarded } P; \text{guarded } Q \rrbracket \implies \Psi \triangleright !P \sim_s !Q$$

2.3.4 Structural equivalence – Theorem 11

- Theorem *bisimCasePushRes*:

$$x \sharp \text{map fst } Cs \implies \Psi \triangleright (\nu x)(\text{Cases } Cs) \sim \text{Cases map } (\lambda(\varphi, P). (\varphi, (\nu x)P)) Cs$$

- Theorem *bisimInputPushRes*:

$$\llbracket x \sharp M; x \sharp \text{vec}; x \sharp N \rrbracket \implies \Psi \triangleright (\nu x)(M(\lambda^* \text{vec } N).P) \sim M(\lambda^* \text{vec } N).(\nu x)P$$

- Theorem *bisimOutputPushRes*:

$$\llbracket x \# M; x \# N \rrbracket \Longrightarrow \Psi \triangleright (\nu x)(M\langle N \rangle.P) \sim M\langle N \rangle.(\nu x)P$$

- Theorem *bisimParAssoc*:

$$\Psi \triangleright P \parallel Q \parallel R \sim P \parallel (Q \parallel R)$$

- Theorem *bisimParComm*:

$$\Psi \triangleright P \parallel Q \sim Q \parallel P$$

- Theorem *bisimResNil*:

$$\Psi \triangleright (\nu x)\mathbf{0} \sim \mathbf{0}$$

- Theorem *bisimScopeExt*:

$$x \# P \Longrightarrow \Psi \triangleright (\nu x)(P \parallel Q) \sim P \parallel (\nu x)Q$$

- Theorem *bisimResComm*:

$$\Psi \triangleright (\nu x)((\nu y)P) \sim (\nu y)((\nu x)P)$$

- Theorem *bangExt*:

$$\text{guarded } P \Longrightarrow \Psi \triangleright !P \sim P \parallel !P$$

- Theorem *bisimParNil*:

$$\Psi \triangleright P \parallel \mathbf{0} \sim P$$

2.3.5 Support of processes in broadcast transitions – Lemma 14

- Theorem *brInputTermSupp*:

$$\Psi \triangleright P \mapsto \iota K(\langle N \rangle) \prec P' \Longrightarrow \text{supp } K \subseteq \text{supp } P$$

- Theorem *brOutputTermSupp*:

$$\Psi \triangleright P \mapsto RBrOut K ((\nu *xvec)\langle N \rangle \prec' P') \Longrightarrow \text{supp } K \subseteq \text{supp } P$$

end

end

References

- [1] Jesper Bengtson. Psi-calculi in Isabelle. *Arch. Formal Proofs*, 2012. https://www.isa-afp.org/entries/Psi_Calculi.html, Formal proof development.
- [2] Jesper Bengtson, Magnus Johansson, Joachim Parrow, and Björn Victor. Psi-calculi: a framework for mobile processes with nominal data and logic. *Log. Methods Comput. Sci.*, 7(1), 2011.
- [3] Jesper Bengtson, Joachim Parrow, and Tjark Weber. Psi-calculi in Isabelle. *J. Autom. Reason.*, 56(1):1–47, 2016.
- [4] Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. Broadcast Psi-calculi with an application to wireless protocols. In Gilles Barthe, Alberto Pardo, and Gerardo Schneider, editors, *Software Engineering and Formal Methods - 9th International Conference, SEFM 2011, Montevideo, Uruguay, November 14-18, 2011. Proceedings*, volume 7041 of *Lecture Notes in Computer Science*, pages 74–89. Springer, 2011.
- [5] Johannes Borgström, Shuqin Huang, Magnus Johansson, Palle Raabjerg, Björn Victor, Johannes Åman Pohjola, and Joachim Parrow. Broadcast psi-calculi with an application to wireless protocols. *Softw. Syst. Model.*, 14(1):201–216, 2015.