

Dynamic Architectures

Diego Marmosler

January 31, 2023

Abstract

The architecture of a system describes the system's overall organization into components and connections between those components. With the emergence of mobile computing, dynamic architectures have become increasingly important. In such architectures, components may appear or disappear, and connections may change over time. In the following we mechanize a theory of dynamic architectures and verify the soundness of a corresponding calculus. Therefore, we first formalize the notion of configuration traces [5] as a model for dynamic architectures. Then, the behavior of single components is formalized in terms of behavior traces and an operator is introduced and studied to extract the behavior of a single component out of a given configuration trace. Then, behavior trace assertions are introduced as a temporal specification technique to specify behavior of components. Reasoning about component behavior in a dynamic context is formalized in terms of a calculus for dynamic architectures [3]. Finally, the soundness of the calculus is verified by introducing an alternative interpretation for behavior trace assertions over configuration traces and proving the rules of the calculus. Since projection may lead to finite as well as infinite behavior traces, they are formalized in terms of coinductive lists. Thus, our theory is based on Lochbihler's [1] formalization of coinductive lists. The theory may be applied to verify properties for dynamic architectures.

Contents

1	A Theory of Dynamic Architectures	4
1.1	Natural Numbers	4
1.2	Extended Natural Numbers	5
1.3	Lazy Lists	5
1.4	Specifying Dynamic Architectures	6
1.4.1	Implication	7
1.4.2	Disjunction	7
1.4.3	Conjunction	7
1.4.4	Negation	8
1.4.5	Quantifiers	8
1.4.6	Atomic Assertions	8
1.4.7	Next Operator	9
1.4.8	Eventually Operator	9
1.4.9	Globally Operator	9
1.4.10	Until Operator	9
1.4.11	Weak Until	9
1.5	Dynamic Components	11
1.6	Projection	12
1.6.1	Monotonicity and Continuity	13
1.6.2	Finiteness	13
1.6.3	Projection not Active	14
1.6.4	Projection Active	15
1.6.5	Same and not Same	17
1.7	Activations	18
1.7.1	Monotonicity and Continuity	19
1.7.2	Not Active	19
1.7.3	Active	20
1.7.4	Same and Not Same	21
1.8	Projection and Activation	23
1.9	Least not Active	25
1.10	Next Active	26
1.11	Latest Activation	29
1.12	Last Activation	30
1.13	Mapping Time Points	31
1.13.1	Configuration Trace to Behavior Trace	32
1.13.2	Behavior Trace to Configuration Trace	34
1.13.3	Relating the Mappings	35
2	A Calculus for Dynamic Architectures	36
2.1	Extended Natural Numbers	36
2.2	Lazy Lists	37
2.3	Dynamic Evaluation of Temporal Operators	37
2.3.1	Simplification Rules	38
2.3.2	No Activations	39
2.4	Specification Operators	39
2.4.1	Predicates	39
2.4.2	True and False	40

2.4.3	Implication	40
2.4.4	Disjunction	42
2.4.5	Conjunction	42
2.4.6	Negation	44
2.4.7	Quantifiers	45
2.4.8	Behavior Assertions	47
2.4.9	Next Operator	51
2.4.10	Eventually Operator	54
2.4.11	Globally Operator	58
2.4.12	Until Operator	62
2.4.13	Weak Until	71

1 A Theory of Dynamic Architectures

The following theory formalizes configuration traces [4, 5] as a model for dynamic architectures. Since configuration traces may be finite as well as infinite, the theory depends on Lochbihler's theory of co-inductive lists [1].

```
theory Configuration-Traces
imports Coinductive.Coinductive-List
begin
```

In the following we first provide some preliminary results for natural numbers, extended natural numbers, and lazy lists. Then, we introduce a locale `@textdynamic_architectures` which introduces basic definitions and corresponding properties for dynamic architectures.

1.1 Natural Numbers

We provide one additional property for natural numbers.

```
lemma boundedGreatest:
  assumes  $P (i::nat)$ 
  and  $\forall n' > n. \neg P n'$ 
  shows  $\exists i' \leq n. P i' \wedge (\forall n'. P n' \longrightarrow n' \leq i')$ 
proof -
  have  $P (i::nat) \implies n \geq i \implies \forall n' > n. \neg P n' \implies (\exists i' \leq n. P i' \wedge (\forall n' \leq n. P n' \longrightarrow n' \leq i'))$ 
  proof (induction n)
  case 0
  then show ?case by auto
next
  case (Suc n)
  then show ?case
  proof cases
  assume  $i = Suc\ n$ 
  then show ?thesis using Suc.prem1s by auto
next
  assume  $\neg(i = Suc\ n)$ 
  thus ?thesis
  proof cases
  assume  $P (Suc\ n)$ 
  thus ?thesis by auto
next
  assume  $\neg P (Suc\ n)$ 
  with Suc.prem1s have  $\forall n' > n. \neg P n'$  using Suc-lessI by blast
  moreover from  $\neg(i = Suc\ n)$  have  $i \leq n$  and  $P\ i$  using Suc.prem1s by auto
  ultimately obtain  $i'$  where  $i' \leq n \wedge P\ i' \wedge (\forall n' \leq n. P\ n' \longrightarrow n' \leq i')$  using Suc.IH by blast
  hence  $i' \leq n$  and  $P\ i'$  and  $(\forall n' \leq n. P\ n' \longrightarrow n' \leq i')$  by auto
  thus ?thesis by (metis le-SucI le-Suc-eq)
  qed
  qed
qed
moreover have  $n \geq i$ 
proof (rule ccontr)
  assume  $\neg(n \geq i)$ 
  hence  $n < i$  by arith
  thus False using assms by blast
qed
ultimately obtain  $i'$  where  $i' \leq n$  and  $P\ i'$  and  $\forall n' \leq n. P\ n' \longrightarrow n' \leq i'$  using assms by blast
```

with *assms* have $\forall n'. P n' \longrightarrow n' \leq i'$ using *not-le-imp-less* by *blast*
 with $\langle i' \leq n \rangle$ and $\langle P i' \rangle$ show *?thesis* by *auto*
 qed

1.2 Extended Natural Numbers

We provide one simple property for the *strict* order over extended natural numbers.

lemma *enat-min*:

assumes $m \geq \text{enat } n'$
 and $\text{enat } n < m - \text{enat } n'$
 shows $\text{enat } n + \text{enat } n' < m$
 using *assms* by (*metis add.commute enat.simps(3) enat-add-mono enat-add-sub-same le-iff-add*)

1.3 Lazy Lists

In the following we provide some additional notation and properties for lazy lists.

notation *LNil* ($[\]_l$)

notation *LCons* (**infixl** $\#_l$ 60)

notation *lappend* (**infixl** $@_l$ 60)

lemma *lnth-lappend[simp]*:

assumes *lfinite xs*
 and $\neg \text{lnull } ys$
 shows $\text{lnth } (xs @_l ys) (\text{the-enat } (\text{llength } xs)) = \text{lhd } ys$

proof –

from *assms* have $\exists k. \text{llength } xs = \text{enat } k$ using *lfinite-conv-llength-enat* by *auto*

then obtain *k* where $\text{llength } xs = \text{enat } k$ by *blast*

hence $\text{lnth } (xs @_l ys) (\text{the-enat } (\text{llength } xs)) = \text{lnth } ys 0$

using *lnth-lappend2[of xs k k ys]* by *simp*

with *assms* show *?thesis* using *lnth-0-conv-lhd* by *simp*

qed

lemma *lfilter-ltake*:

assumes $\forall (n::\text{nat}) \leq \text{llength } xs. n \geq i \longrightarrow (\neg P (\text{lnth } xs n))$

shows $\text{lfilter } P xs = \text{lfilter } P (\text{ltake } i xs)$

proof –

have $\text{lfilter } P xs = \text{lfilter } P ((\text{ltake } i xs) @_l (\text{ldrop } i xs))$

using *lappend-ltake-ldrop[of (enat i) xs]* by *simp*

hence $\text{lfilter } P xs = (\text{lfilter } P ((\text{ltake } i) xs)) @_l (\text{lfilter } P (\text{ldrop } i xs))$ by *simp*

show *?thesis*

proof cases

assume $\text{enat } i \leq \text{llength } xs$

have $\forall x < \text{llength } (\text{ldrop } i xs). \neg P (\text{lnth } (\text{ldrop } i xs) x)$

proof (*rule allI*)

fix *x* show $\text{enat } x < \text{llength } (\text{ldrop } (\text{enat } i) xs) \longrightarrow \neg P (\text{lnth } (\text{ldrop } (\text{enat } i) xs) x)$

proof

assume $\text{enat } x < \text{llength } (\text{ldrop } (\text{enat } i) xs)$

moreover have $\text{llength } (\text{ldrop } (\text{enat } i) xs) = \text{llength } xs - \text{enat } i$

using *llength-ldrop[of enat i]* by *simp*

ultimately have $\text{enat } x < \text{llength } xs - \text{enat } i$ by *simp*

with $\langle \text{enat } i \leq \text{llength } xs \rangle$ have $\text{enat } x + \text{enat } i < \text{llength } xs$

using *enat-min[of i llength xs x]* by *simp*

moreover have $\text{enat } i + \text{enat } x = \text{enat } x + \text{enat } i$ by *simp*

ultimately have $enat\ i + enat\ x < llength\ xs$ **by** *arith*
hence $i + x < llength\ xs$ **by** *simp*
hence $lnth\ (ldrop\ i\ xs)\ x = lnth\ xs\ (x + the-enat\ i)$ **using** *lnth-ldrop[of enat i x xs]* **by** *simp*
moreover have $x + i \geq i$ **by** *simp*
with *assms* $\langle i + x < llength\ xs \rangle$ **have** $\neg P\ (lnth\ xs\ (x + the-enat\ i))$
by (*simp add: assms(1) add.commute*)
ultimately show $\neg P\ (lnth\ (ldrop\ i\ xs)\ x)$ **using** *assms* **by** *simp*
qed
qed
hence $lfilter\ P\ (ldrop\ i\ xs) = []_l$ **by** (*metis diverge-lfilter-LNil in-lset-conv-lnth*)
with $\langle lfilter\ P\ xs = (lfilter\ P\ ((ltake\ i)\ xs)) @_l (lfilter\ P\ (ldrop\ i\ xs)) \rangle$
show $lfilter\ P\ xs = lfilter\ P\ (ltake\ i\ xs)$ **by** *simp*
next
assume $\neg enat\ i \leq llength\ xs$
hence $enat\ i > llength\ xs$ **by** *simp*
hence $ldrop\ i\ xs = []_l$ **by** *simp*
hence $lfilter\ P\ (ldrop\ i\ xs) = []_l$ **using** *lfilter-LNil[of P]* **by** *arith*
with $\langle lfilter\ P\ xs = (lfilter\ P\ ((ltake\ i)\ xs)) @_l (lfilter\ P\ (ldrop\ i\ xs)) \rangle$
show $lfilter\ P\ xs = lfilter\ P\ (ltake\ i\ xs)$ **by** *simp*
qed
qed

lemma *lfilter-lfinite[simp]*:
assumes *lfinite* $(lfilter\ P\ t)$
and $\neg lfinite\ t$
shows $\exists n. \forall n' > n. \neg P\ (lnth\ t\ n')$
proof –
from *assms* **have** $finite\ \{n. enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\}$ **using** *lfinite-lfilter* **by** *auto*
then obtain k
where *sset*: $\{n. enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\} \subseteq \{n. n < k \wedge enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\}$
using *finite-nat-bounded*[of $\{n. enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\}$] **by** *auto*
show *?thesis*
proof (*rule ccontr*)
assume $\neg(\exists n. \forall n' > n. \neg P\ (lnth\ t\ n'))$
hence $\forall n. \exists n' > n. P\ (lnth\ t\ n')$ **by** *simp*
then obtain n' **where** $n' > k$ **and** $P\ (lnth\ t\ n')$ **by** *auto*
moreover from $\langle \neg lfinite\ t \rangle$ **have** $n' < llength\ t$ **by** (*simp add: not-lfinite-llength*)
ultimately have $n' \notin \{n. n < k \wedge enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\}$ **and**
 $n' \in \{n. enat\ n < llength\ t \wedge P\ (lnth\ t\ n)\}$ **by** *auto*
with *sset* **show** *False* **by** *auto*
qed
qed

1.4 Specifying Dynamic Architectures

In the following we formalize dynamic architectures in terms of configuration traces, i.e., sequences of architecture configurations. Moreover, we introduce definitions for operations to support the specification of configuration traces.

typedecl *cnf*
type-synonym $trace = nat \Rightarrow cnf$
consts *arch:: trace set*

type-synonym $cta = trace \Rightarrow nat \Rightarrow bool$

1.4.1 Implication

definition $imp :: cta \Rightarrow cta \Rightarrow cta$ (**infixl** \longrightarrow^c 10)
where $\gamma \longrightarrow^c \gamma' \equiv \lambda t n. \gamma t n \longrightarrow \gamma' t n$

declare $imp-def[simp]$

lemma $impI[intro!]$:
fixes $t n$
assumes $\gamma t n \Longrightarrow \gamma' t n$
shows $(\gamma \longrightarrow^c \gamma') t n$ **using** *assms* **by** *simp*

lemma $impE[elim!]$:
fixes $t n$
assumes $(\gamma \longrightarrow^c \gamma') t n$ **and** $\gamma t n$ **and** $\gamma' t n \Longrightarrow \gamma'' t n$
shows $\gamma'' t n$ **using** *assms* **by** *simp*

1.4.2 Disjunction

definition $disj :: cta \Rightarrow cta \Rightarrow cta$ (**infixl** \vee^c 15)
where $\gamma \vee^c \gamma' \equiv \lambda t n. \gamma t n \vee \gamma' t n$

declare $disj-def[simp]$

lemma $disjI1[intro]$:
assumes $\gamma t n$
shows $(\gamma \vee^c \gamma') t n$ **using** *assms* **by** *simp*

lemma $disjI2[intro]$:
assumes $\gamma' t n$
shows $(\gamma \vee^c \gamma') t n$ **using** *assms* **by** *simp*

lemma $disjE[elim!]$:
assumes $(\gamma \vee^c \gamma') t n$
and $\gamma t n \Longrightarrow \gamma'' t n$
and $\gamma' t n \Longrightarrow \gamma'' t n$
shows $\gamma'' t n$ **using** *assms* **by** *auto*

1.4.3 Conjunction

definition $conj :: cta \Rightarrow cta \Rightarrow cta$ (**infixl** \wedge^c 20)
where $\gamma \wedge^c \gamma' \equiv \lambda t n. \gamma t n \wedge \gamma' t n$

declare $conj-def[simp]$

lemma $conjI[intro!]$:
fixes n
assumes $\gamma t n$ **and** $\gamma' t n$
shows $(\gamma \wedge^c \gamma') t n$ **using** *assms* **by** *simp*

lemma $conjE[elim!]$:
fixes n
assumes $(\gamma \wedge^c \gamma') t n$ **and** $\gamma t n \Longrightarrow \gamma' t n \Longrightarrow \gamma'' t n$
shows $\gamma'' t n$ **using** *assms* **by** *simp*

1.4.4 Negation

definition $neg :: cta \Rightarrow cta$ (\neg^c - [19] 19)
where $\neg^c \gamma \equiv \lambda t n. \neg \gamma t n$

declare $neg-def[simp]$

lemma $negI[intro!]$:
assumes $\gamma t n \Longrightarrow False$
shows $(\neg^c \gamma) t n$ using *assms* by *auto*

lemma $negE[elim!]$:
assumes $(\neg^c \gamma) t n$
and $\gamma t n$
shows $\gamma' t n$ using *assms* by *simp*

1.4.5 Quantifiers

definition $all :: ('a \Rightarrow cta) \Rightarrow cta$ (**binder** \forall_c 10)
where $all P \equiv \lambda t n. (\forall y. (P y t n))$

declare $all-def[simp]$

lemma $allI[intro!]$:
assumes $\bigwedge x. \gamma x t n$
shows $(\forall_c x. \gamma x) t n$ using *assms* by *simp*

lemma $allE[elim!]$:
fixes n
assumes $(\forall_c x. \gamma x) t n$ and $\gamma x t n \Longrightarrow \gamma' t n$
shows $\gamma' t n$ using *assms* by *simp*

definition $ex :: ('a \Rightarrow cta) \Rightarrow cta$ (**binder** \exists_c 10)
where $ex P \equiv \lambda t n. (\exists y. (P y t n))$

declare $ex-def[simp]$

lemma $exI[intro!]$:
assumes $\gamma x t n$
shows $(\exists_c x. \gamma x) t n$ using *assms* *HOL.exI* by *simp*

lemma $exE[elim!]$:
assumes $(\exists_c x. \gamma x) t n$ and $\bigwedge x. \gamma x t n \Longrightarrow \gamma' t n$
shows $\gamma' t n$ using *assms* *HOL.exE* by *auto*

1.4.6 Atomic Assertions

First we provide rules for basic behavior assertions.

definition $ca :: (cnf \Rightarrow bool) \Rightarrow cta$
where $ca \varphi \equiv \lambda t n. \varphi (t n)$

lemma $caI[intro!]$:
fixes n
assumes $\varphi (t n)$

shows $(ca \ \varphi) \ t \ n$ using *assms ca-def by simp*

lemma *caE[elim]*:

fixes n

assumes $(ca \ \varphi) \ t \ n$

shows $\varphi \ (t \ n)$ using *assms ca-def by simp*

1.4.7 Next Operator

definition *next* :: $cta \Rightarrow cta$ ($\circ_c(-)$ 24)

where $\circ_c(\gamma) \equiv \lambda(t::(nat \Rightarrow cnf)) \ n. \ \gamma \ t \ (Suc \ n)$

1.4.8 Eventually Operator

definition *evt* :: $cta \Rightarrow cta$ ($\diamond_c(-)$ 23)

where $\diamond_c(\gamma) \equiv \lambda(t::(nat \Rightarrow cnf)) \ n. \ \exists n' \geq n. \ \gamma \ t \ n'$

1.4.9 Globally Operator

definition *glob* :: $cta \Rightarrow cta$ ($\square_c(-)$ 22)

where $\square_c(\gamma) \equiv \lambda(t::(nat \Rightarrow cnf)) \ n. \ \forall n' \geq n. \ \gamma \ t \ n'$

lemma *globI[intro!]*:

fixes n'

assumes $\forall n \geq n'. \ \gamma \ t \ n$

shows $(\square_c(\gamma)) \ t \ n'$ using *assms glob-def by simp*

lemma *globE[elim!]*:

fixes $n \ n'$

assumes $(\square_c(\gamma)) \ t \ n$ and $n' \geq n$

shows $\gamma \ t \ n'$ using *assms glob-def by simp*

1.4.10 Until Operator

definition *until* :: $cta \Rightarrow cta \Rightarrow cta$ (**infixl** \mathfrak{U}_c 21)

where $\gamma' \ \mathfrak{U}_c \ \gamma \equiv \lambda(t::(nat \Rightarrow cnf)) \ n. \ \exists n'' \geq n. \ \gamma \ t \ n'' \ \wedge \ (\forall n' \geq n. \ n' < n'' \longrightarrow \gamma' \ t \ n')$

lemma *untilI[intro]*:

fixes n

assumes $\exists n'' \geq n. \ \gamma \ t \ n'' \ \wedge \ (\forall n' \geq n. \ n' < n'' \longrightarrow \gamma' \ t \ n')$

shows $(\gamma' \ \mathfrak{U}_c \ \gamma) \ t \ n$ using *assms until-def by simp*

lemma *untilE[elim]*:

fixes n

assumes $(\gamma' \ \mathfrak{U}_c \ \gamma) \ t \ n$

shows $\exists n'' \geq n. \ \gamma \ t \ n'' \ \wedge \ (\forall n' \geq n. \ n' < n'' \longrightarrow \gamma' \ t \ n')$ using *assms until-def by simp*

1.4.11 Weak Until

definition *wuntil* :: $cta \Rightarrow cta \Rightarrow cta$ (**infixl** \mathfrak{W}_c 20)

where $\gamma' \ \mathfrak{W}_c \ \gamma \equiv \gamma' \ \mathfrak{U}_c \ \gamma \ \vee^c \ \square_c(\gamma')$

lemma *wUntilI[intro]*:

fixes n

assumes $(\exists n'' \geq n. \ \gamma \ t \ n'' \ \wedge \ (\forall n' \geq n. \ n' < n'' \longrightarrow \gamma' \ t \ n')) \ \vee \ (\forall n' \geq n. \ \gamma' \ t \ n')$

shows $(\gamma' \ \mathfrak{W}_c \ \gamma) \ t \ n$ using *assms wuntil-def by auto*

lemma *wUntilE[elim]*:

fixes $n\ n'$

assumes $(\gamma' \mathfrak{W}_c \gamma) t\ n$

shows $(\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n')) \vee (\forall n' \geq n. \gamma' t\ n')$

proof –

from *assms* have $(\gamma' \mathfrak{U}_c \gamma \vee^c \Box_c(\gamma')) t\ n$ using *wuntil-def* by *simp*

hence $(\gamma' \mathfrak{U}_c \gamma) t\ n \vee (\Box_c(\gamma')) t\ n$ by *simp*

thus *?thesis*

proof

assume $(\gamma' \mathfrak{U}_c \gamma) t\ n$

hence $\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n')$ by *auto*

thus *?thesis* by *auto*

next

assume $(\Box_c \gamma') t\ n$

hence $\forall n' \geq n. \gamma' t\ n'$ by *auto*

thus *?thesis* by *auto*

qed

qed

lemma *wUntil-Glob*:

assumes $(\gamma' \mathfrak{W}_c \gamma) t\ n$

and $(\Box_c(\gamma' \longrightarrow^c \gamma'')) t\ n$

shows $(\gamma'' \mathfrak{W}_c \gamma) t\ n$

proof

from *assms(1)* have $(\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n')) \vee (\forall n' \geq n. \gamma' t\ n')$ using *wUntilE* by *simp*

thus $(\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma'' t\ n')) \vee (\forall n' \geq n. \gamma'' t\ n')$

proof

assume $\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n')$

show $(\exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma'' t\ n')) \vee (\forall n' \geq n. \gamma'' t\ n')$

proof –

from $\langle \exists n'' \geq n. \gamma t\ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n') \rangle$ obtain n'' where $n'' \geq n$ and $\gamma t\ n''$ and *a1*: $\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n'$ by *auto*

moreover have $\forall n' \geq n. n' < n'' \longrightarrow \gamma'' t\ n'$

proof

fix n'

show $n' \geq n \longrightarrow n' < n'' \longrightarrow \gamma'' t\ n'$

proof (rule *HOL.impI*[*OF HOL.impI*])

assume $n' \geq n$ and $n' < n''$

with *assms(2)* have $(\gamma' \longrightarrow^c \gamma'') t\ n'$ using *globE* by *simp*

hence $\gamma' t\ n' \longrightarrow \gamma'' t\ n'$ using *impE* by *auto*

moreover from *a1* $\langle n' \geq n \rangle \langle n' < n'' \rangle$ have $\gamma' t\ n'$ by *simp*

ultimately show $\gamma'' t\ n'$ by *simp*

qed

qed

ultimately show *?thesis* by *auto*

qed

next

assume *a1*: $\forall n' \geq n. \gamma' t\ n'$

have $\forall n' \geq n. \gamma'' t\ n'$

proof

fix n'

show $n' \geq n \longrightarrow \gamma'' t\ n'$

proof

```

assume  $n' \geq n$ 
with assms(2) have  $(\gamma' \longrightarrow^c \gamma'') t n'$  using globE by simp
hence  $\gamma' t n' \longrightarrow \gamma'' t n'$  using impE by auto
moreover from a1  $\langle n' \geq n \rangle$  have  $\gamma' t n'$  by simp
ultimately show  $\gamma'' t n'$  by simp
qed
qed
thus  $(\exists n'' \geq n. \gamma t n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma'' t n')) \vee (\forall n' \geq n. \gamma'' t n')$  by simp
qed
qed

```

1.5 Dynamic Components

To support the specification of patterns over dynamic architectures we provide a locale for dynamic components. It takes the following type parameters:

- *id*: a type for component identifiers
- *cmp*: a type for components
- *cnf*: a type for architecture configurations

```

locale dynamic-component =
  fixes tCMP :: 'id  $\Rightarrow$  cnf  $\Rightarrow$  'cmp ( $\sigma$ _-) [0,110]60)
  and active :: 'id  $\Rightarrow$  cnf  $\Rightarrow$  bool ( $\|$ _- $\|$ _- [0,110]60)
begin

```

The locale requires two parameters:

- *tCMP* is an operator to obtain a component with a certain identifier from an architecture configuration.
- *active* is a predicate to assert whether a certain component is activated within an architecture configuration.

The locale provides some general properties about its parameters and introduces six important operators over configuration traces:

- An operator to extract the behavior of a certain component out of a given configuration trace.
- An operator to obtain the number of activations of a certain component within a given configuration trace.
- An operator to obtain the least point in time (before a certain point in time) from which on a certain component is not activated anymore.
- An operator to obtain the latest point in time where a certain component was activated.
- Two operators to map time-points between configuration traces and behavior traces.

Moreover, the locale provides several properties about the operators and their relationships.

```

lemma nact-active:
  fixes t::nat  $\Rightarrow$  cnf

```

and $n::nat$
and n''
and id
assumes $\|id\|_t n$
and $n'' \geq n$
and $\neg (\exists n' \geq n. n' < n'' \wedge \|id\|_t n')$
shows $n=n''$
using *assms le-eq-less-or-eq* **by** *auto*

lemma *nact-exists*:

fixes $t::nat \Rightarrow cnf$
assumes $\exists i \geq n. \|c\|_t i$
shows $\exists i \geq n. \|c\|_t i \wedge (\nexists k. n \leq k \wedge k < i \wedge \|c\|_t k)$

proof –

let $?L = LEAST i. (i \geq n \wedge \|c\|_t i)$
from *assms* **have** $?L \geq n \wedge \|c\|_t ?L$ **using** *LeastI*[of $\lambda x::nat. (x \geq n \wedge \|c\|_t x)$] **by** *auto*
moreover **have** $\nexists k. n \leq k \wedge k < ?L \wedge \|c\|_t k$ **using** *not-less-Least* **by** *auto*
ultimately show *?thesis* **by** *blast*

qed

lemma *lActive-least*:

assumes $\exists i \geq n. i < llength\ t \wedge \|c\|_{lnth\ t\ i}$
shows $\exists i \geq n. (i < llength\ t \wedge \|c\|_{lnth\ t\ i} \wedge (\nexists k. n \leq k \wedge k < i \wedge k < llength\ t \wedge \|c\|_{lnth\ t\ k}))$

proof –

let $?L = LEAST i. (i \geq n \wedge i < llength\ t \wedge \|c\|_{lnth\ t\ i})$
from *assms* **have** $?L \geq n \wedge ?L < llength\ t \wedge \|c\|_{lnth\ t\ ?L}$
using *LeastI*[of $\lambda x::nat. (x \geq n \wedge x < llength\ t \wedge \|c\|_{lnth\ t\ x})$] **by** *auto*
moreover **have** $\nexists k. n \leq k \wedge k < llength\ t \wedge k < ?L \wedge \|c\|_{lnth\ t\ k}$ **using** *not-less-Least* **by** *auto*
ultimately show *?thesis* **by** *blast*

qed

1.6 Projection

In the following we introduce an operator which extracts the behavior of a certain component out of a given configuration trace.

definition *proj*:: $'id \Rightarrow (cnf\ llist) \Rightarrow ('cmp\ llist) (\pi_-(-) [0,110]60)$
where $proj\ c = lmap (\lambda cnf. (\sigma_c(cnf))) \circ (lfilter (active\ c))$

lemma *proj-lnil* [*simp,intro*]:

$\pi_c(\llbracket l \rrbracket) = \llbracket l \rrbracket$ **using** *proj-def* **by** *simp*

lemma *proj-lnull* [*simp*]:

$\pi_c(t) = \llbracket l \rrbracket \longleftrightarrow (\forall k \in lset\ t. \neg \|c\|_k)$

proof

assume $\pi_c(t) = \llbracket l \rrbracket$
hence $lfilter (active\ c)\ t = \llbracket l \rrbracket$ **using** *proj-def lmap-eq-LNil* **by** *auto*
thus $\forall k \in lset\ t. \neg \|c\|_k$ **using** *lfilter-eq-LNil*[of *active c*] **by** *simp*

next

assume $\forall k \in lset\ t. \neg \|c\|_k$
hence $lfilter (active\ c)\ t = \llbracket l \rrbracket$ **by** *simp*
thus $\pi_c(t) = \llbracket l \rrbracket$ **using** *proj-def* **by** *simp*

qed

lemma *proj-LCons* [*simp*]:

$\pi_i(x \#_l xs) = (if\ \|i\|_x\ then\ (\sigma_i(x)) \#_l (\pi_i(xs))\ else\ \pi_i(xs))$

using *proj-def* by *simp*

lemma *proj-llength[simp]*:
 llength ($\pi_c(t)$) \leq *llength* *t*
 using *llength-lfilter-ile proj-def* by *simp*

lemma *proj-ltake*:
 assumes $\forall (n'::nat) \leq \text{llength } t. n' \geq n \longrightarrow (\neg \|c\|_{\text{lnth } t } n')$
 shows $\pi_c(t) = \pi_c(\text{ltake } n \ t)$ using *lfilter-ltake proj-def* *assms* by (*metis comp-apply*)

lemma *proj-finite-bound*:
 assumes *lfinite* ($\pi_c(\text{inf-llist } t)$)
 shows $\exists n. \forall n' > n. \neg \|c\|_{t } n'$
 using *assms lfilter-lfinite[of active c inf-llist t] proj-def* by *simp*

1.6.1 Monotonicity and Continuity

lemma *proj-mcont*:
 shows *mcont lSup lprefix lSup lprefix* (*proj c*)
proof –
 have *mcont lSup lprefix lSup lprefix* ($\lambda x. \text{lmap } (\lambda \text{cnf}. \sigma_c(\text{cnf})) (\text{lfilter } (\text{active } c) \ x)$)
 by *simp*
 moreover have $(\lambda x. \text{lmap } (\lambda \text{cnf}. \sigma_c(\text{cnf})) (\text{lfilter } (\text{active } c) \ x)) =$
 $\text{lmap } (\lambda \text{cnf}. \sigma_c(\text{cnf})) \circ \text{lfilter } (\text{active } c)$ by *auto*
 ultimately show *?thesis* using *proj-def* by *simp*
qed

lemma *proj-mcont2mcont*:
 assumes *mcont lub ord lSup lprefix f*
 shows *mcont lub ord lSup lprefix* ($\lambda x. \pi_c(f \ x)$)
proof –
 have *mcont lSup lprefix lSup lprefix* (*proj c*) using *proj-mcont* by *simp*
 with *assms* show *?thesis* using *lprefix.mcont2mcont[of lSup lprefix proj c]* by *simp*
qed

lemma *proj-mono-prefix[simp]*:
 assumes *lprefix t t'*
 shows *lprefix* ($\pi_c(t)$) ($\pi_c(t')$)
proof –
 from *assms* have *lprefix* (*lfilter* (*active c*) *t*) (*lfilter* (*active c*) *t'*) using *lprefix-lfilterI* by *simp*
 hence *lprefix* ($\text{lmap } (\lambda \text{cnf}. \sigma_c(\text{cnf})) (\text{lfilter } (\text{active } c) \ t)$)
 ($\text{lmap } (\lambda \text{cnf}. \sigma_c(\text{cnf})) (\text{lfilter } (\text{active } c) \ t')$) using *lmap-lprefix* by *simp*
 thus *?thesis* using *proj-def* by *simp*
qed

1.6.2 Finiteness

lemma *proj-finite[simp]*:
 assumes *lfinite t*
 shows *lfinite* ($\pi_c(t)$)
 using *assms proj-def* by *simp*

lemma *proj-finite2*:
 assumes $\forall (n'::nat) \leq \text{llength } t. n' \geq n \longrightarrow (\neg \|c\|_{\text{lnth } t } n')$
 shows *lfinite* ($\pi_c(t)$) using *assms proj-ltake proj-finite* by *simp*

lemma *proj-append-lfinite[simp]*:

fixes $t\ t'$
assumes *lfinite* t
shows $\pi_c(t @_l t') = (\pi_c(t)) @_l (\pi_c(t'))$ (**is** *?lhs=?rhs*)

proof –

have $?lhs = (lmap (\lambda cnf. \sigma_c(cnf)) \circ (lfilter (active\ c))) (t @_l t')$ **using** *proj-def* **by** *simp*
also have $\dots = lmap (\lambda cnf. \sigma_c(cnf)) (lfilter (active\ c) (t @_l t'))$ **by** *simp*
also from *assms* **have** $\dots = lmap (\lambda cnf. \sigma_c(cnf))$
 $((lfilter (active\ c) t) @_l (lfilter (active\ c) t'))$ **by** *simp*
also have $\dots = (@_l) (lmap (\lambda cnf. \sigma_c(cnf)) (lfilter (active\ c) t))$
 $(lmap (\lambda cnf. \sigma_c(cnf)) (lfilter (active\ c) t'))$ **using** *lmap-lappend-distrib* **by** *simp*
also have $\dots = ?rhs$ **using** *proj-def* **by** *simp*
finally show *?thesis* .

qed

lemma *proj-one*:

assumes $\exists i. i < llength\ t \wedge \|c\|_{lnth\ t\ i}$
shows $llength\ (\pi_c(t)) \geq 1$

proof –

from *assms* **have** $\exists x \in lset\ t. \|c\|_x$ **using** *lset-conv-lnth* **by** *force*
hence $\neg lfilter (\lambda k. \|c\|_k) t = []$ **using** *lfilter-eq-LNil[of (\lambda k. \|c\|_k)]* **by** *blast*
hence $\neg \pi_c(t) = []$ **using** *proj-def* **by** *fastforce*
thus *?thesis* **by** (*simp add: ileI1 lnull-def one-eSuc*)

qed

1.6.3 Projection not Active

lemma *proj-not-active[simp]*:

assumes *enat* $n < llength\ t$
and $\neg \|c\|_{lnth\ t\ n}$
shows $\pi_c(ltake (Suc\ n) t) = \pi_c(ltake\ n\ t)$ (**is** *?lhs = ?rhs*)

proof –

from *assms* **have** $ltake (enat\ (Suc\ n)) t = (ltake (enat\ n) t) @_l ((lnth\ t\ n) \#_l [])$
using *ltake-Suc-conv-snoc-lnth* **by** *blast*
hence $?lhs = \pi_c((ltake (enat\ n) t) @_l ((lnth\ t\ n) \#_l []))$ **by** *simp*
moreover have $\dots = (\pi_c(ltake (enat\ n) t)) @_l (\pi_c((lnth\ t\ n) \#_l []))$ **by** *simp*
moreover from *assms* **have** $\pi_c((lnth\ t\ n) \#_l []) = []$ **by** *simp*
ultimately show *?thesis* **by** *simp*

qed

lemma *proj-not-active-same*:

assumes *enat* $n \leq (n'::enat)$
and $\neg lfinite\ t \vee n'-1 < llength\ t$
and $\nexists k. k \geq n \wedge k < n' \wedge k < llength\ t \wedge \|c\|_{lnth\ t\ k}$
shows $\pi_c(ltake\ n'\ t) = \pi_c(ltake\ n\ t)$

proof –

have $\pi_c(ltake (n + (n' - n)) t) = \pi_c((ltake\ n\ t) @_l (ltake (n'-n) (ldrop\ n\ t)))$
by (*simp add: ltake-plus-conv-lappend*)
hence $\pi_c(ltake (n + (n' - n)) t) =$
 $(\pi_c(ltake\ n\ t)) @_l (\pi_c(ltake (n'-n) (ldrop\ n\ t)))$ **by** *simp*
moreover have $\pi_c(ltake (n'-n) (ldrop\ n\ t)) = []$

proof –

have $\forall k \in \{lnth (ltake (n' - enat\ n) (ldrop (enat\ n) t))\ na \mid$
 $na. enat\ na < llength (ltake (n' - enat\ n) (ldrop (enat\ n) t))\}. \neg \|c\|_k$

proof

fix k **assume** $k \in \{lnth (ltake (n' - enat\ n) (ldrop (enat\ n) t))\ na \mid$

$na. \text{ enat } na < \text{llength } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t))\}$
then obtain k' **where** $\text{enat } k' < \text{llength } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t))$
and $k = \text{lnth } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \ k'$ **by** *auto*
have $\text{enat } (k' + n) < \text{llength } t$
proof –
from $\langle \text{enat } k' < \text{llength } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \rangle$ **have** $\text{enat } k' < n' - n$ **by** *simp*
hence $\text{enat } k' + n < n'$ **using** *assms(1) enat-min* **by** *auto*
show *?thesis*
proof *cases*
assume *lfinite t*
with $\langle \neg \text{lfinite } t \vee n' - 1 < \text{llength } t \rangle$ **have** $n' - 1 < \text{llength } t$ **by** *simp*
hence $n' < \text{eSuc } (\text{llength } t)$ **by** (*metis eSuc-minus-1 enat-minus-mono1 leD leI*)
hence $n' \leq \text{llength } t$ **using** *eSuc-ile-mono ileI1* **by** *blast*
with $\langle \text{enat } k' + n < n' \rangle$ **show** *?thesis* **by** (*simp add: add.commute*)
next
assume $\neg \text{lfinite } t$
hence $\text{llength } t = \infty$ **using** *not-lfinite-llength* **by** *auto*
thus *?thesis* **by** *simp*
qed
qed
moreover **have** $k = \text{lnth } t \ (k' + n)$
proof –
from $\langle \text{enat } k' < \text{llength } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \rangle$
have $\text{enat } k' < n' - \text{enat } n$ **by** *auto*
hence $\text{lnth } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \ k' = \text{lnth } (\text{ldrop } (\text{enat } n) t) \ k'$
using *lnth-ltake[of k' n' - enat n]* **by** *simp*
with $\langle \text{enat } (k' + n) < \text{llength } t \rangle$ **show** *?thesis* **using** *lnth-ldrop[of n k' t]*
using $\langle k = \text{lnth } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \ k' \rangle$ **by** (*simp add: add.commute*)
qed
moreover **from** $\langle \text{enat } n \leq (n'::\text{enat}) \rangle$ **have** $k' + \text{the-enat } n \geq n$ **by** *auto*
moreover **from** $\langle \text{enat } k' < \text{llength } (\text{ltake } (n' - \text{enat } n) (\text{ldrop } (\text{enat } n) t)) \rangle$ **have** $k' + n < n'$
using *assms(1) enat-min* **by** *auto*
ultimately **show** $\neg \|c\|_k$ **using** $\langle \#k. k \geq n \wedge k < n' \wedge k < \text{llength } t \wedge \|c\|_{\text{lnth } t \ k} \rangle$ **by** *simp*
qed
hence $\forall k \in \text{lset } (\text{ltake } (n' - n) (\text{ldrop } n \ t)). \neg \|c\|_k$
using *lset-conv-lnth[of (ltake (n' - enat n) (ldrop (enat n) t))]* **by** *simp*
thus *?thesis* **using** *proj-lnull* **by** *auto*
qed
moreover **from** *assms* **have** $n + (n' - n) = n'$
by (*meson enat.distinct(1) enat-add-sub-same enat-diff-cancel-left enat-le-plus-same(1) less-imp-le*)
ultimately **show** *?thesis* **by** *simp*
qed

1.6.4 Projection Active

lemma *proj-active[simp]*:

assumes $\text{enat } i < \text{llength } t \ \|c\|_{\text{lnth } t \ i}$
shows $\pi_c(\text{ltake } (\text{Suc } i) \ t) = (\pi_c(\text{ltake } i \ t)) \ @_i \ ((\sigma_c(\text{lnth } t \ i)) \ #_i \ []_i)$ (**is** *?lhs = ?rhs*)

proof –

from *assms* **have** $\text{ltake } (\text{enat } (\text{Suc } i)) \ t = (\text{ltake } (\text{enat } i) \ t) \ @_i \ ((\text{lnth } t \ i) \ #_i \ []_i)$
using *ltake-Suc-conv-snoc-lnth* **by** *blast*
hence $?lhs = \pi_c((\text{ltake } (\text{enat } i) \ t) \ @_i \ ((\text{lnth } t \ i) \ #_i \ []_i))$ **by** *simp*
moreover **have** $\dots = (\pi_c(\text{ltake } (\text{enat } i) \ t)) \ @_i \ (\pi_c((\text{lnth } t \ i) \ #_i \ []_i))$ **by** *simp*
moreover **from** *assms* **have** $\pi_c((\text{lnth } t \ i) \ #_i \ []_i) = (\sigma_c(\text{lnth } t \ i)) \ #_i \ []_i$ **by** *simp*
ultimately **show** *?thesis* **by** *simp*

qed

lemma *proj-active-append*:

assumes $a1: (n::nat) \leq i$

and $a2: \text{enat } i < (n'::\text{enat})$

and $a3: \neg \text{lfinite } t \vee n'-1 < \text{llength } t$

and $a4: \|c\|_{\text{lnth } t} i$

and $\forall i'. (n \leq i' \wedge \text{enat } i' < n' \wedge i' < \text{llength } t \wedge \|c\|_{\text{lnth } t} i') \longrightarrow (i' = i)$

shows $\pi_c(\text{ltake } n' t) = (\pi_c(\text{ltake } n t)) \text{@}_l ((\sigma_c(\text{lnth } t i)) \#_l \llbracket i \rrbracket_l)$ (is ?lhs = ?rhs)

proof –

have ?lhs = $\pi_c(\text{ltake } (\text{Suc } i) t)$

proof –

from $a2$ have $\text{Suc } i \leq n'$ by (simp add: *Suc-ile-eq*)

moreover from $a3$ have $\neg \text{lfinite } t \vee n'-1 < \text{llength } t$ by *simp*

moreover have $\nexists k. \text{enat } k \geq \text{enat } (\text{Suc } i) \wedge k < n' \wedge k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k$

proof

assume $\exists k. \text{enat } k \geq \text{enat } (\text{Suc } i) \wedge k < n' \wedge k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k$

then obtain k where $\text{enat } k \geq \text{enat } (\text{Suc } i)$ and $k < n'$ and $k < \text{llength } t$ and $\|c\|_{\text{lnth } t} k$ by *blast*

moreover from $\langle \text{enat } k \geq \text{enat } (\text{Suc } i) \rangle$ have $\text{enat } k \geq n$

using *assms* by (meson *dual-order.trans enat-ord-simps(1) le-SucI*)

ultimately have $\text{enat } k = \text{enat } i$ using *assms* using *enat-ord-simps(1)* by *blast*

with $\langle \text{enat } k \geq \text{enat } (\text{Suc } i) \rangle$ show *False* by *simp*

qed

ultimately show ?thesis using *proj-not-active-same[of Suc i n' t c]* by *simp*

qed

also have $\dots = (\pi_c(\text{ltake } i t)) \text{@}_l ((\sigma_c(\text{lnth } t i)) \#_l \llbracket i \rrbracket_l)$

proof –

have $i < \text{llength } t$

proof cases

assume *lfinite t*

with $a3$ have $n'-1 < \text{llength } t$ by *simp*

hence $n' \leq \text{llength } t$ by (*metis eSuc-minus-1 enat-minus-mono1 ileI1 not-le*)

with $a2$ show $\text{enat } i < \text{llength } t$ by *simp*

next

assume $\neg \text{lfinite } t$

thus ?thesis by (*metis enat-ord-code(4) llength-eq-infty-conv-lfinite*)

qed

with $a4$ show ?thesis by *simp*

qed

also have $\dots = ?rhs$

proof –

from $a1$ have $\text{enat } n \leq \text{enat } i$ by *simp*

moreover from $a2$ $a3$ have $\neg \text{lfinite } t \vee \text{enat } i-1 < \text{llength } t$

using *enat-minus-mono1 less-imp-le order.strict-trans1* by *blast*

moreover have $\nexists k. k \geq n \wedge \text{enat } k < \text{enat } i \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k$

proof

assume $\exists k. k \geq n \wedge \text{enat } k < \text{enat } i \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k$

then obtain k where $k \geq n$ and $\text{enat } k < \text{enat } i$ and $\text{enat } k < \text{llength } t$ and $\|c\|_{\text{lnth } t} k$ by *blast*

moreover from $\langle \text{enat } k < \text{enat } i \rangle$ have $\text{enat } k < n'$ using *assms dual-order.strict-trans* by *blast*

ultimately have $\text{enat } k = \text{enat } i$ using *assms* by *simp*

with $\langle \text{enat } k < \text{enat } i \rangle$ show *False* by *simp*

qed

ultimately show ?thesis using *proj-not-active-same[of n i t c]* by *simp*

qed

finally show ?thesis by *simp*

qed

1.6.5 Same and not Same

lemma *proj-same-not-active*:

assumes $n \leq n'$
 and $\text{enat } (n'-1) < \text{llength } t$
 and $\pi_c(\text{ltake } n' t) = \pi_c(\text{ltake } n t)$
 shows $\nexists k. k \geq n \wedge k < n' \wedge \|c\|_{\text{lnth } t} k$

proof

assume $\exists k. k \geq n \wedge k < n' \wedge \|c\|_{\text{lnth } t} k$
 then obtain i where $i \geq n$ and $i < n'$ and $\|c\|_{\text{lnth } t} i$ by *blast*
 moreover from $\langle \text{enat } (n'-1) < \text{llength } t \rangle$ and $\langle i < n' \rangle$ have $i < \text{llength } t$
 by (*metis diff-Suc-1 dual-order.strict-trans enat-ord-simps(2) lessE*)
 ultimately have $\pi_c(\text{ltake } (\text{Suc } i) t) =$
 $(\pi_c(\text{ltake } i t)) @_i ((\sigma_c(\text{lnth } t i)) \#_i []_i)$ by *simp*
 moreover from $\langle i < n' \rangle$ have $\text{Suc } i \leq n'$ by *simp*
 hence $\text{lprefix}(\pi_c(\text{ltake } (\text{Suc } i) t)) (\pi_c(\text{ltake } n' t))$ by *simp*
 then obtain tl where $\pi_c(\text{ltake } n' t) = (\pi_c(\text{ltake } (\text{Suc } i) t)) @_i tl$
 using *lprefix-conv-lappend* by *auto*
 moreover from $\langle n \leq i \rangle$ have $\text{lprefix}(\pi_c(\text{ltake } n t)) (\pi_c(\text{ltake } i t))$ by *simp*
 hence $\text{lprefix}(\pi_c(\text{ltake } n t)) (\pi_c(\text{ltake } i t))$ by *simp*
 then obtain hd where $\pi_c(\text{ltake } i t) = (\pi_c(\text{ltake } n t)) @_i hd$
 using *lprefix-conv-lappend* by *auto*
 ultimately have $\pi_c(\text{ltake } n' t) =$
 $((\pi_c(\text{ltake } n t)) @_i hd) @_i ((\sigma_c(\text{lnth } t i)) \#_i []_i) @_i tl$ by *simp*
 also have $\dots = ((\pi_c(\text{ltake } n t)) @_i hd) @_i ((\sigma_c(\text{lnth } t i)) \#_i tl)$
 using *lappend-snocL1-conv-LCons2*[of $(\pi_c(\text{ltake } n t)) @_i hd \sigma_c(\text{lnth } t i)$] by *simp*
 also have $\dots = (\pi_c(\text{ltake } n t)) @_i (hd @_i ((\sigma_c(\text{lnth } t i)) \#_i tl))$
 using *lappend-assoc* by *auto*
 also have $\pi_c(\text{ltake } n' t) = (\pi_c(\text{ltake } n' t)) @_i []_i$ by *simp*
 finally have $(\pi_c(\text{ltake } n' t)) @_i []_i = (\pi_c(\text{ltake } n t)) @_i (hd @_i ((\sigma_c(\text{lnth } t i)) \#_i tl))$.
 moreover from *assms(3)* have $\text{llength } (\pi_c(\text{ltake } n' t)) = \text{llength } (\pi_c(\text{ltake } n t))$ by *simp*
 ultimately have $\text{lfinite } (\pi_c(\text{ltake } n' t)) \longrightarrow []_i = hd @_i ((\sigma_c(\text{lnth } t i)) \#_i tl)$
 using *assms(3) lappend-eq-lappend-conv*[of $\pi_c(\text{ltake } n' t) \pi_c(\text{ltake } n t) []_i$] by *simp*
 moreover have $\text{lfinite } (\pi_c(\text{ltake } n' t))$ by *simp*
 ultimately have $[]_i = hd @_i ((\sigma_c(\text{lnth } t i)) \#_i tl)$ by *simp*
 hence $(\sigma_c(\text{lnth } t i)) \#_i tl = []_i$ using *LNil-eq-lappend-iff* by *auto*
 thus *False* by *simp*

qed

lemma *proj-not-same-active*:

assumes $\text{enat } n \leq (n'::\text{enat})$
 and $(\neg \text{lfinite } t) \vee n'-1 < \text{llength } t$
 and $\neg(\pi_c(\text{ltake } n' t) = \pi_c(\text{ltake } n t))$
 shows $\exists k. k \geq n \wedge k < n' \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k$

proof (*rule ccontr*)

assume $\neg(\exists k. k \geq n \wedge k < n' \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k)$
 have $\pi_c(\text{ltake } n' t) = \pi_c(\text{ltake } (\text{enat } n) t)$

proof *cases*

assume *lfinite t*
 hence $\text{llength } t \neq \infty$ by (*simp add: lfinite-llength-enat*)
 hence $\text{enat } (\text{the-enat } (\text{llength } t)) = \text{llength } t$ by *auto*
 with *assms* $\langle \neg(\exists k \geq n. k < n' \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k) \rangle$
 show *?thesis* using *proj-not-active-same*[of $n n' t c$] by *simp*

next

assume $\neg \text{lfinite } t$
 with *assms* $\langle \neg(\exists k \geq n. k < n' \wedge \text{enat } k < \text{llength } t \wedge \|c\|_{\text{lnth } t} k) \rangle$

show *?thesis* **using** *proj-not-active-same*[of n n' t c] **by** *simp*
qed
with *assms* **show** *False* **by** *simp*
qed

1.7 Activations

We also introduce an operator to obtain the number of activations of a certain component within a given configuration trace.

definition $nAct :: 'id \Rightarrow enat \Rightarrow (cnf\ llist) \Rightarrow enat \langle (- \# -) \rangle$ **where**
 $\langle c \#_n t \rangle \equiv llength (\pi_c (ltake\ n\ t))$

lemma $nAct-0$ [*simp*]:
 $\langle c \#_0 t \rangle = 0$ **by** (*simp add: nAct-def*)

lemma $nAct-NIL$ [*simp*]:
 $\langle c \#_n [] \rangle = 0$ **by** (*simp add: nAct-def*)

lemma $nAct-Null$:
assumes $llength\ t \geq n$
and $\langle c \#_n t \rangle = 0$
shows $\forall i < n. \neg \|c\|_{lnth\ t\ i}$

proof –

from *assms* **have** $lnull (\pi_c (ltake\ n\ t))$ **using** $nAct-def\ lnull-def$ **by** *simp*

hence $\pi_c (ltake\ n\ t) = []$ **using** $lnull-def$ **by** *blast*

hence $(\forall k \in lset (ltake\ n\ t). \neg \|c\|_k)$ **by** *simp*

show *?thesis*

proof (*rule ccontr*)

assume $\neg (\forall i < n. \neg \|c\|_{lnth\ t\ i})$

then obtain i **where** $i < n$ **and** $\|c\|_{lnth\ t\ i}$ **by** *blast*

moreover have $enat\ i < llength (ltake\ n\ t) \wedge lnth (ltake\ n\ t)\ i = (lnth\ t\ i)$

proof

from $\langle llength\ t \geq n \rangle$ **have** $n = \min\ n\ (llength\ t)$ **using** *min.orderE* **by** *auto*

hence $llength (ltake\ n\ t) = n$ **by** *simp*

with $\langle i < n \rangle$ **show** $enat\ i < llength (ltake\ n\ t)$ **by** *auto*

from $\langle i < n \rangle$ **show** $lnth (ltake\ n\ t)\ i = (lnth\ t\ i)$ **using** $lnth-ltake$ **by** *auto*

qed

hence $(lnth\ t\ i \in lset (ltake\ n\ t))$ **using** $in-lset-conv-lnth$ [of $lnth\ t\ i\ ltake\ n\ t$] **by** *blast*

ultimately show *False* **using** $\langle (\forall k \in lset (ltake\ n\ t). \neg \|c\|_k) \rangle$ **by** *simp*

qed

qed

lemma $nAct-ge-one$ [*simp*]:
assumes $llength\ t \geq n$
and $i < n$
and $\|c\|_{lnth\ t\ i}$
shows $\langle c \#_n t \rangle \geq enat\ 1$

proof (*rule ccontr*)

assume $\neg (\langle c \#_n t \rangle \geq enat\ 1)$

hence $\langle c \#_n t \rangle < enat\ 1$ **by** *simp*

hence $\langle c \#_n t \rangle < 1$ **using** $enat-1$ **by** *simp*

hence $\langle c \#_n t \rangle = 0$ **using** $Suc-ile-eq \langle \neg enat\ 1 \leq \langle c \#_n t \rangle \rangle$ $zero-enat-def$ **by** *auto*

with $\langle llength\ t \geq n \rangle$ **have** $\forall i < n. \neg \|c\|_{lnth\ t\ i}$ **using** $nAct-Null$ **by** *simp*

with *assms* **show** *False* **by** *simp*

qed

lemma *nAct-finite[simp]*:

assumes $n \neq \infty$

shows $\exists n'. \langle c \#_n t \rangle = \text{enat } n'$

proof –

from *assms* **have** *lfinite* (*ltake* n t) **by** *simp*

hence *lfinite* ($\pi_c(\text{ltake } n \ t)$) **by** *simp*

hence $\exists n'. \text{llength } (\pi_c(\text{ltake } n \ t)) = \text{enat } n'$ **using** *lfinite-llength-enat*[*of* $\pi_c(\text{ltake } n \ t)$] **by** *simp*

thus *?thesis* **using** *nAct-def* **by** *simp*

qed

lemma *nAct-enat-the-nat[simp]*:

assumes $n \neq \infty$

shows $\text{enat } (\text{the-enat } (\langle c \#_n t \rangle)) = \langle c \#_n t \rangle$

proof –

from *assms* **have** $\langle c \#_n t \rangle \neq \infty$ **by** *simp*

thus *?thesis* **using** *enat-the-enat* **by** *simp*

qed

1.7.1 Monotonicity and Continuity

lemma *nAct-mcont*:

shows *mcont* *lSup* *lprefix* *Sup* (\leq) (*nAct* c n)

proof –

have *mcont* *lSup* *lprefix* *lSup* *lprefix* (*ltake* n) **by** *simp*

hence *mcont* *lSup* *lprefix* *lSup* *lprefix* ($\lambda t. \pi_c(\text{ltake } n \ t)$)

using *proj-mcont2mcont*[*of* *lSup* *lprefix* (*ltake* n)] **by** *simp*

hence *mcont* *lSup* *lprefix* *Sup* (\leq) ($\lambda t. \text{llength } (\pi_c(\text{ltake } n \ t))$) **by** *simp*

moreover **have** *nAct* c $n = (\lambda t. \text{llength } (\pi_c(\text{ltake } n \ t)))$ **using** *nAct-def* **by** *auto*
ultimately show *?thesis* **by** *simp*

qed

lemma *nAct-mono*:

assumes $n \leq n'$

shows $\langle c \#_n t \rangle \leq \langle c \#_{n'} t \rangle$

proof –

from *assms* **have** *lprefix* (*ltake* n t) (*ltake* n' t) **by** *simp*

hence *lprefix* ($\pi_c(\text{ltake } n \ t)$) ($\pi_c(\text{ltake } n' \ t)$) **by** *simp*

hence $\text{llength } (\pi_c(\text{ltake } n \ t)) \leq \text{llength } (\pi_c(\text{ltake } n' \ t))$

using *lprefix-llength-le*[*of* ($\pi_c(\text{ltake } n \ t)$)] **by** *simp*

thus *?thesis* **using** *nAct-def* **by** *simp*

qed

lemma *nAct-strict-mono-back*:

assumes $\langle c \#_n t \rangle < \langle c \#_{n'} t \rangle$

shows $n < n'$

proof (*rule ccontr*)

assume $\neg n < n'$

hence $n \geq n'$ **by** *simp*

hence $\langle c \#_n t \rangle \geq \langle c \#_{n'} t \rangle$ **using** *nAct-mono* **by** *simp*

thus *False* **using** *assms* **by** *simp*

qed

1.7.2 Not Active

lemma *nAct-not-active[simp]*:

fixes $n::nat$
and $n'::nat$
and $t::(cnf\ llist)$
and $c::'id$
assumes $enat\ i < llength\ t$
and $\neg \|c\|_{lnth\ t\ i}$
shows $\langle c \#_{Suc\ i}\ t \rangle = \langle c \#_i\ t \rangle$
proof –
from *assms* **have** $\pi_c(ltake\ (Suc\ i)\ t) = \pi_c(ltake\ i\ t)$ **by** *simp*
hence $llength\ (\pi_c(ltake\ (enat\ (Suc\ i))\ t)) = llength\ (\pi_c(ltake\ i\ t))$ **by** *simp*
moreover **have** $llength\ (\pi_c(ltake\ i\ t)) \neq \infty$
using *length-eq-infty-conv-lfinite*[of $\pi_c(ltake\ (enat\ i)\ t)$] **by** *simp*
ultimately **have** $llength\ (\pi_c(ltake\ (Suc\ i)\ t)) = llength\ (\pi_c(ltake\ i\ t))$
using *the-enat-eSuc* **by** *simp*
with *nAct-def* **show** *?thesis* **by** *simp*
qed

lemma *nAct-not-active-same*:
assumes $enat\ n \leq (n'::enat)$
and $n'-1 < llength\ t$
and $\nexists k. enat\ k \geq n \wedge k < n' \wedge \|c\|_{lnth\ t\ k}$
shows $\langle c \#_{n'}\ t \rangle = \langle c \#_n\ t \rangle$
using *assms proj-not-active-same nAct-def* **by** *simp*

1.7.3 Active

lemma *nAct-active*[*simp*]:
fixes $n::nat$
and $n'::nat$
and $t::(cnf\ llist)$
and $c::'id$
assumes $enat\ i < llength\ t$
and $\|c\|_{lnth\ t\ i}$
shows $\langle c \#_{Suc\ i}\ t \rangle = eSuc\ (\langle c \#_i\ t \rangle)$
proof –
from *assms* **have** $\pi_c(ltake\ (Suc\ i)\ t) =$
 $(\pi_c(ltake\ i\ t)) \ @_i\ ((\sigma_c(lnth\ t\ i)) \ #_i\ []_i)$ **by** *simp*
hence $llength\ (\pi_c(ltake\ (enat\ (Suc\ i))\ t)) = eSuc\ (llength\ (\pi_c(ltake\ i\ t)))$
using *plus-1-eSuc one-eSuc* **by** *simp*
moreover **have** $llength\ (\pi_c(ltake\ i\ t)) \neq \infty$
using *length-eq-infty-conv-lfinite*[of $\pi_c(ltake\ (enat\ i)\ t)$] **by** *simp*
ultimately **have** $llength\ (\pi_c(ltake\ (Suc\ i)\ t)) = eSuc\ (llength\ (\pi_c(ltake\ i\ t)))$
using *the-enat-eSuc* **by** *simp*
with *nAct-def* **show** *?thesis* **by** *simp*
qed

lemma *nAct-active-suc*:
fixes $n::nat$
and $n'::enat$
and $t::(cnf\ llist)$
and $c::'id$
assumes $\neg\ lfinite\ t \vee n'-1 < llength\ t$
and $n \leq i$
and $enat\ i < n'$
and $\|c\|_{lnth\ t\ i}$
and $\forall i'. (n \leq i' \wedge enat\ i' < n' \wedge i' < llength\ t \wedge \|c\|_{lnth\ t\ i'}) \longrightarrow (i' = i)$

shows $\langle c \#_{n'} t \rangle = eSuc (\langle c \#_n t \rangle)$
proof –
from *assms* **have** $\pi_c(\text{ltake } n' t) = (\pi_c(\text{ltake } (enat\ n) t)) @_l ((\sigma_c(\text{lnth } t\ i)) \#_l [])$
using *proj-active-append[of n i n' t c]* **by** *blast*
moreover **have** $\text{llength } ((\pi_c(\text{ltake } (enat\ n) t)) @_l ((\sigma_c(\text{lnth } t\ i)) \#_l [])) =$
 $eSuc (\text{llength } (\pi_c(\text{ltake } (enat\ n) t)))$ **using** *one-eSuc eSuc-plus-1* **by** *simp*
ultimately **show** *?thesis* **using** *nAct-def* **by** *simp*
qed

lemma *nAct-less*:

assumes $enat\ k < \text{llength } t$
and $n \leq k$
and $k < (n'::enat)$
and $\|c\|_{\text{lnth } t\ k}$
shows $\langle c \#_n t \rangle < \langle c \#_{n'} t \rangle$

proof –

have $\langle c \#_k t \rangle \neq \infty$ **by** *simp*
then **obtain** *en* **where** *en-def*: $\langle c \#_k t \rangle = enat\ en$ **by** *blast*
moreover **have** $eSuc (enat\ en) \leq \langle c \#_{n'} t \rangle$

proof –

from *assms* **have** $Suc\ k \leq n'$ **using** *Suc-ile-eq* **by** *simp*
hence $\langle c \#_{Suc\ k} t \rangle \leq \langle c \#_{n'} t \rangle$ **using** *nAct-mono* **by** *simp*
moreover **from** *assms* **have** $\langle c \#_{Suc\ k} t \rangle = eSuc (\langle c \#_k t \rangle)$ **by** *simp*
ultimately **have** $eSuc (\langle c \#_k t \rangle) \leq \langle c \#_{n'} t \rangle$ **by** *simp*
thus *?thesis* **using** *en-def* **by** *simp*

qed

moreover **have** $enat\ en < eSuc (enat\ en)$ **by** *simp*
ultimately **have** $enat\ en < \langle c \#_{n'} t \rangle$ **using** *less-le-trans[of enat en eSuc (enat en)]* **by** *simp*
moreover **have** $\langle c \#_n t \rangle \leq enat\ en$

proof –

from *assms* **have** $\langle c \#_n t \rangle \leq \langle c \#_k t \rangle$ **using** *nAct-mono* **by** *simp*
thus *?thesis* **using** *en-def* **by** *simp*

qed

ultimately **show** *?thesis* **using** *le-less-trans[of < c #_n t >]* **by** *simp*

qed

lemma *nAct-less-active*:

assumes $n' - 1 < \text{llength } t$
and $\langle c \#_{enat\ n} t \rangle < \langle c \#_{n'} t \rangle$
shows $\exists i \geq n. i < n' \wedge \|c\|_{\text{lnth } t\ i}$

proof (*rule ccontr*)

assume $\neg (\exists i \geq n. i < n' \wedge \|c\|_{\text{lnth } t\ i})$
moreover **have** $enat\ n \leq n'$ **using** *assms(2) less-imp-le nAct-strict-mono-back* **by** *blast*
ultimately **have** $\langle c \#_n t \rangle = \langle c \#_{n'} t \rangle$ **using** $\langle n' - 1 < \text{llength } t \rangle$ *nAct-not-active-same* **by** *simp*
thus *False* **using** *assms* **by** *simp*

qed

1.7.4 Same and Not Same

lemma *nAct-same-not-active*:

assumes $\langle c \#_{n'} \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$
shows $\forall k \geq n. k < n' \longrightarrow \neg \|c\|_t k$

proof (*rule ccontr*)

assume $\neg (\forall k \geq n. k < n' \longrightarrow \neg \|c\|_t k)$
then **obtain** *k* **where** $k \geq n$ **and** $k < n'$ **and** $\|c\|_t k$ **by** *blast*
hence $\langle c \#_{Suc\ k} \text{inf-llist } t \rangle = eSuc (\langle c \#_k \text{inf-llist } t \rangle)$ **by** *simp*

moreover have $\langle c \#_k \text{inf-llist } t \rangle \neq \infty$ by *simp*
 ultimately have $\langle c \#_k \text{inf-llist } t \rangle < \langle c \#_{\text{Suc } k} \text{inf-llist } t \rangle$ by *fastforce*
 moreover from $\langle n \leq k \rangle$ have $\langle c \#_n \text{inf-llist } t \rangle \leq \langle c \#_k \text{inf-llist } t \rangle$ using *nAct-mono* by *simp*
 moreover from $\langle k < n' \rangle$ have $\text{Suc } k \leq n'$ by (*simp add: Suc-ile-eq*)
 hence $\langle c \#_{\text{Suc } k} \text{inf-llist } t \rangle \leq \langle c \#_{n'} \text{inf-llist } t \rangle$ using *nAct-mono* by *simp*
 ultimately show *False* using *assms* by *simp*
 qed

lemma *nAct-not-same-active*:

assumes $\langle c \#_{\text{enat } n} t \rangle < \langle c \#_{n'} t \rangle$
 and $\neg \text{lfinite } t \vee n' - 1 < \text{llength } t$
 shows $\exists (i::\text{nat}) \geq n. \text{enat } i < n' \wedge i < \text{llength } t \wedge \|c\|_{\text{Inth } t} i$

proof –

from *assms* have $\text{llength}(\pi_c(\text{ltake } n \ t)) < \text{llength}(\pi_c(\text{ltake } n' \ t))$ using *nAct-def* by *simp*
 hence $\pi_c(\text{ltake } n' \ t) \neq \pi_c(\text{ltake } n \ t)$ by *auto*
 moreover from *assms* have $\text{enat } n < n'$ using *nAct-strict-mono-back*[of $c \ \text{enat } n \ t \ n'$] by *simp*
 ultimately show *?thesis* using *proj-not-same-active*[of $n \ n' \ t \ c$] *assms* by *simp*
 qed

lemma *nAct-less-llength-active*:

assumes $x < \text{llength}(\pi_c(t))$
 and $\text{enat } x = \langle c \#_{\text{enat } n'} t \rangle$
 shows $\exists (i::\text{nat}) \geq n'. i < \text{llength } t \wedge \|c\|_{\text{Inth } t} i$

proof –

have $\text{llength}(\pi_c(\text{ltake } n' \ t)) < \text{llength}(\pi_c(t))$ using *assms(1) assms(2) nAct-def* by *auto*
 hence $\text{llength}(\pi_c(\text{ltake } n' \ t)) < \text{llength}(\pi_c(\text{ltake } (\text{llength } t) \ t))$ by (*simp add: ltake-all*)
 hence $\langle c \#_{\text{enat } n'} t \rangle < \langle c \#_{\text{llength } t} t \rangle$ using *nAct-def* by *simp*
 moreover have $\neg \text{lfinite } t \vee \text{llength } t - 1 < \text{llength } t$
 proof (rule *Meson.imp-to-disjD[OF HOL.impI]*)

assume *lfinite t*

hence $\text{llength } t \neq \infty$ by (*simp add: llength-eq-infty-conv-lfinite*)

moreover have $\text{llength } t > 0$

proof –

from $\langle x < \text{llength}(\pi_c(t)) \rangle$ have $\text{llength}(\pi_c(t)) > 0$ by *auto*

thus *?thesis* using *proj-llength order.strict-trans2* by *blast*

qed

ultimately show $\text{llength } t - 1 < \text{llength } t$ by (*metis One-nat-def lfinite t diff-Suc-less enat-ord-simps(2) idiff-enat-enat lfinite-conv-llength-enat one-enat-def zero-enat-def*)

qed

ultimately show *?thesis* using *nAct-not-same-active*[of $c \ n' \ t \ \text{llength } t$] by *simp*

qed

lemma *nAct-exists*:

assumes $x < \text{llength}(\pi_c(t))$
 shows $\exists (n'::\text{nat}). \text{enat } x = \langle c \#_{n'} t \rangle$

proof –

have $x < \text{llength}(\pi_c(t)) \longrightarrow (\exists (n'::\text{nat}). \text{enat } x = \langle c \#_{n'} t \rangle)$

proof (*induction x*)

case 0

thus *?case* by (*metis nAct-0 zero-enat-def*)

next

case (*Suc x*)

show *?case*

proof

assume $\text{Suc } x < \text{llength}(\pi_c(t))$

hence $x < \text{length } (\pi_c(t))$ **using** *Suc-ile-eq less-imp-le* **by** *auto*
with *Suc.IH* **obtain** n' **where** $\text{enat } x = \langle c \#_{\text{enat } n'} t \rangle$ **by** *blast*
with $\langle x < \text{length } (\pi_c(t)) \rangle$ **have** $\exists i \geq n'. i < \text{length } t \wedge \|c\|_{\text{Inth } t} i$
using *nAct-less-llength-active[of x c t n']* **by** *simp*
then obtain i **where** $i \geq n'$ **and** $i < \text{length } t$ **and** $\|c\|_{\text{Inth } t} i$
and $\nexists k. n' \leq k \wedge k < i \wedge k < \text{length } t \wedge \|c\|_{\text{Inth } t} k$ **using** *lActive-least[of n' t c]* **by** *auto*
moreover from $\langle i < \text{length } t \rangle$ **have** $\neg \text{lfinite } t \vee \text{enat } (\text{Suc } i) - 1 < \text{length } t$
by (*simp add: one-enat-def*)
moreover have $\text{enat } i < \text{enat } (\text{Suc } i)$ **by** *simp*
moreover have $\forall i'. (n' \leq i' \wedge \text{enat } i' < \text{enat } (\text{Suc } i) \wedge i' < \text{length } t \wedge \|c\|_{\text{Inth } t} i') \longrightarrow (i' = i)$
proof (*rule HOL.impI[THEN HOL.allI]*)
fix i' **assume** $n' \leq i' \wedge \text{enat } i' < \text{enat } (\text{Suc } i) \wedge i' < \text{length } t \wedge \|c\|_{\text{Inth } t} i'$
with $\langle \nexists k. n' \leq k \wedge k < i \wedge k < \text{length } t \wedge \|c\|_{\text{Inth } t} k \rangle$ **show** $i' = i$ **by** *fastforce*
qed
ultimately have $\langle c \#_{\text{Suc } i} t \rangle = \text{eSuc } (\langle c \#_{n'} t \rangle)$ **using** *nAct-active-suc[of t Suc i n' i c]* **by** *simp*
with $\langle \text{enat } x = \langle c \#_{\text{enat } n'} t \rangle \rangle$ **have** $\langle c \#_{\text{Suc } i} t \rangle = \text{eSuc } (\text{enat } x)$ **by** *simp*
thus $\exists n'. \text{enat } (\text{Suc } x) = \langle c \#_{\text{enat } n'} t \rangle$ **by** (*metis eSuc-enat*)
qed
qed
with *assms* **show** *?thesis* **by** *simp*
qed

1.8 Projection and Activation

In the following we provide some properties about the relationship between the projection and activations operator.

lemma *nAct-le-proj*:

$\langle c \#_n t \rangle \leq \text{length } (\pi_c(t))$

proof –

from *nAct-def* **have** $\langle c \#_n t \rangle = \text{length } (\pi_c(\text{ltake } n t))$ **by** *simp*

moreover have $\text{length } (\pi_c(\text{ltake } n t)) \leq \text{length } (\pi_c(t))$

proof –

have *lprefix* $(\text{ltake } n t) t$ **by** *simp*

hence *lprefix* $(\pi_c(\text{ltake } n t)) (\pi_c(t))$ **by** *simp*

hence $\text{length } (\pi_c(\text{ltake } n t)) \leq \text{length } (\pi_c(t))$ **using** *lprefix-llength-le* **by** *blast*

thus *?thesis* **by** *auto*

qed

thus *?thesis* **using** *nAct-def* **by** *simp*

qed

lemma *proj-nAct*:

assumes $\text{enat } n < \text{length } t$

shows $\pi_c(\text{ltake } n t) = \text{ltake } (\langle c \#_n t \rangle) (\pi_c(t))$ (**is** *?lhs = ?rhs*)

proof –

have *?lhs* = $\text{ltake } (\text{length } (\pi_c(\text{ltake } n t))) (\pi_c(\text{ltake } n t))$

using *ltake-all[of $\pi_c(\text{ltake } n t)$ $\text{length } (\pi_c(\text{ltake } n t))$]* **by** *simp*

also have $\dots = \text{ltake } (\text{length } (\pi_c(\text{ltake } n t))) ((\pi_c(\text{ltake } n t)) @_l (\pi_c(\text{ldrop } n t)))$

using *ltake-lappend1[of $\text{length } (\pi_c(\text{ltake } (\text{enat } n) t))$ $\pi_c(\text{ltake } n t)$ $(\pi_c(\text{ldrop } n t))$]* **by** *simp*

also have $\dots = \text{ltake } (\langle c \#_n t \rangle) ((\pi_c(\text{ltake } n t)) @_l (\pi_c(\text{ldrop } n t)))$ **using** *nAct-def* **by** *simp*

also have $\dots = \text{ltake } (\langle c \#_n t \rangle) (\pi_c((\text{ltake } (\text{enat } n) t) @_l (\text{ldrop } n t)))$ **by** *simp*

also have $\dots = \text{ltake } (\langle c \#_n t \rangle) (\pi_c(t))$ **using** *lappend-ltake-ldrop[of n t]* **by** *simp*

finally show *?thesis* **by** *simp*

qed

lemma *proj-active-nth*:

assumes $enat (Suc\ i) < llength\ t \parallel c \parallel_{lnth\ t\ i}$
shows $lnth\ (\pi_c(t))\ (the-enat\ (\langle c\ \#_i\ t \rangle)) = \sigma_c(lnth\ t\ i)$
proof –
from *assms* **have** $enat\ i < llength\ t$ **using** *Suc-ile-eq*[of $i\ llength\ t$] **by** *auto*
with *assms* **have** $\pi_c(ltake\ (Suc\ i)\ t) = (\pi_c(ltake\ i\ t))\ @_l\ ((\sigma_c(lnth\ t\ i))\ \#_l\ [])$ **by** *simp*
moreover **have** $lnth\ ((\pi_c(ltake\ i\ t))\ @_l\ ((\sigma_c(lnth\ t\ i))\ \#_l\ []))$
 $(the-enat\ (llength\ (\pi_c(ltake\ i\ t)))) = \sigma_c(lnth\ t\ i)$
proof –
have $\neg\ lnull\ ((\sigma_c(lnth\ t\ i))\ \#_l\ [])$ **by** *simp*
moreover **have** $lfinite\ (\pi_c(ltake\ i\ t))$ **by** *simp*
ultimately **have** $lnth\ ((\pi_c(ltake\ i\ t))\ @_l\ ((\sigma_c(lnth\ t\ i))\ \#_l\ []))$
 $(the-enat\ (llength\ (\pi_c(ltake\ i\ t)))) = lhd\ ((\sigma_c(lnth\ t\ i))\ \#_l\ [])$ **by** *simp*
also **have** $\dots = \sigma_c(lnth\ t\ i)$ **by** *simp*
finally **show** $lnth\ ((\pi_c(ltake\ i\ t))\ @_l\ ((\sigma_c(lnth\ t\ i))\ \#_l\ []))$
 $(the-enat\ (llength\ (\pi_c(ltake\ i\ t)))) = \sigma_c(lnth\ t\ i)$ **by** *simp*
qed
ultimately **have** $\sigma_c(lnth\ t\ i) = lnth\ (\pi_c(ltake\ (Suc\ i)\ t))$
 $(the-enat\ (llength\ (\pi_c(ltake\ i\ t))))$ **by** *simp*
also **have** $\dots = lnth\ (\pi_c(ltake\ (Suc\ i)\ t))\ (the-enat\ (\langle c\ \#_i\ t \rangle))$ **using** *nAct-def* **by** *simp*
also **have** $\dots = lnth\ (ltake\ (\langle c\ \#_{Suc\ i}\ t \rangle)\ (\pi_c(t)))\ (the-enat\ (\langle c\ \#_i\ t \rangle))$
using *proj-nAct*[of $Suc\ i\ t\ c$] *assms* **by** *simp*
also **have** $\dots = lnth\ (\pi_c(t))\ (the-enat\ (\langle c\ \#_i\ t \rangle))$
proof –
from *assms* **have** $\langle c\ \#_{Suc\ i}\ t \rangle = eSuc\ (\langle c\ \#_i\ t \rangle)$ **using** $\langle enat\ i < llength\ t \rangle$ **by** *simp*
moreover **have** $\langle c\ \#_i\ t \rangle < eSuc\ (\langle c\ \#_i\ t \rangle)$ **using** *iless-Suc-eq*[of $the-enat\ (\langle c\ \#_{enat\ i}\ t \rangle)$] **by** *simp*
ultimately **have** $\langle c\ \#_i\ t \rangle < (\langle c\ \#_{Suc\ i}\ t \rangle)$ **by** *simp*
hence $enat\ (the-enat\ (\langle c\ \#_{Suc\ i}\ t \rangle)) > enat\ (the-enat\ (\langle c\ \#_i\ t \rangle))$ **by** *simp*
thus *?thesis* **using** *lnth-ltake*[of $the-enat\ (\langle c\ \#_i\ t \rangle)\ the-enat\ (\langle c\ \#_{enat\ (Suc\ i)}\ t \rangle)\ \pi_c(t)$] **by** *simp*
qed
finally **show** *?thesis* ..
qed

lemma *nAct-eq-proj*:

assumes $\neg(\exists i \geq n. \parallel c \parallel_{lnth\ t\ i})$
shows $\langle c\ \#_n\ t \rangle = llength\ (\pi_c(t))$ (**is** *?lhs* = *?rhs*)

proof –

from *nAct-def* **have** $?lhs = llength\ (\pi_c(ltake\ n\ t))$ **by** *simp*
moreover **from** *assms* **have** $\forall (n'::nat) \leq llength\ t. n' \geq n \longrightarrow (\neg \parallel c \parallel_{lnth\ t\ n'})$ **by** *simp*
hence $\pi_c(t) = \pi_c(ltake\ n\ t)$ **using** *proj-ltake* **by** *simp*
ultimately **show** *?thesis* **by** *simp*

qed

lemma *nAct-llength-proj*:

assumes $\exists i \geq n. \parallel c \parallel_{t\ i}$
shows $llength\ (\pi_c(inf-llist\ t)) \geq eSuc\ (\langle c\ \#_n\ inf-llist\ t \rangle)$

proof –

from $\langle \exists i \geq n. \parallel c \parallel_{t\ i} \rangle$ **obtain** i **where** $i \geq n$ **and** $\parallel c \parallel_{t\ i}$
and $\neg(\exists k \geq n. k < i \wedge k < llength\ (inf-llist\ t) \wedge \parallel c \parallel_{t\ k})$
using *lActive-least*[of $n\ inf-llist\ t\ c$] **by** *auto*
moreover **have** $llength\ (\pi_c(inf-llist\ t)) \geq \langle c\ \#_{Suc\ i}\ inf-llist\ t \rangle$ **using** *nAct-le-proj* **by** *simp*
moreover **have** $eSuc\ (\langle c\ \#_n\ inf-llist\ t \rangle) = \langle c\ \#_{Suc\ i}\ inf-llist\ t \rangle$

proof –

have $enat\ (Suc\ i) < llength\ (inf-llist\ t)$ **by** *simp*
moreover **have** $i < Suc\ i$ **by** *simp*
moreover **from** $\langle \neg(\exists k \geq n. k < i \wedge k < llength\ (inf-llist\ t) \wedge \parallel c \parallel_{t\ k}) \rangle$

have $\forall i'. n \leq i' \wedge i' < \text{Suc } i \wedge \|c\|_{\text{lnth } (\text{inf-list } t) i'} \longrightarrow i' = i$ **by** *fastforce*
ultimately show *?thesis* **using** *nAct-active-suc* $\langle i \geq n \rangle \langle \|c\|_t i \rangle$ **by** *simp*
qed
ultimately show *?thesis* **by** *simp*
qed

1.9 Least not Active

In the following, we introduce an operator to obtain the least point in time before a certain point in time where a component was deactivated.

definition *lNAct* :: '*id* \Rightarrow (*nat* \Rightarrow *cnf*) \Rightarrow *nat* \Rightarrow *nat* ($\langle - \Leftarrow - \rangle$)
where $\langle c \Leftarrow t \rangle_n \equiv (\text{LEAST } n'. n = n' \vee (n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k)))$

lemma *lNAct0*[*simp*]:
 $\langle c \Leftarrow t \rangle_0 = 0$
by (*simp add: lNAct-def*)

lemma *lNAct-least*:
assumes $n = n' \vee n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k)$
shows $\langle c \Leftarrow t \rangle_n \leq n'$
using *Least-le*[*of* $\lambda n'. n = n' \vee (n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k))$] *lNAct-def* **using** *assms* **by** *auto*

lemma *lNAct-ex*: $\langle c \Leftarrow t \rangle_n = n \vee \langle c \Leftarrow t \rangle_n < n \wedge (\nexists k. k \geq \langle c \Leftarrow t \rangle_n \wedge k < n \wedge \|c\|_t k)$

proof –

let $?P = \lambda n'. n = n' \vee n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k)$
from *lNAct-def* **have** $\langle c \Leftarrow t \rangle_n = (\text{LEAST } n'. ?P n')$ **by** *simp*
moreover have $?P n$ **by** *simp*
with *LeastI* **have** $?P (\text{LEAST } n'. ?P n')$.
ultimately show *?thesis* **by** *auto*
qed

lemma *lNAct-notActive*:
fixes *c t n k*
assumes $k \geq \langle c \Leftarrow t \rangle_n$
and $k < n$
shows $\neg \|c\|_t k$
by (*metis assms lNAct-ex leD*)

lemma *lNActGe*:
fixes *c t n n'*
assumes $n' \geq \langle c \Leftarrow t \rangle_n$
and $\|c\|_t n'$
shows $n' \geq n$
using *assms lNAct-notActive leI* **by** *blast*

lemma *lNActLe*[*simp*]:
fixes *n n'*
shows $\langle c \Leftarrow t \rangle_n \leq n$
using *lNAct-ex less-or-eq-imp-le* **by** *blast*

lemma *lNActLe-nact*:
fixes *n n'*
assumes $n' = n \vee (n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k))$
shows $\langle c \Leftarrow t \rangle_n \leq n'$

using *assms lNAct-def Least-le*[of $\lambda n'. n=n' \vee (n' < n \wedge (\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k))$]] **by** *auto*

lemma *lNAct-active*:

fixes *cid t n*

assumes $\forall k < n. \|cid\|_t k$

shows $\langle cid \Leftarrow t \rangle_n = n$

using *assms lNAct-ex* **by** *blast*

lemma *nAct-mono-back*:

fixes *c t* **and** *n* **and** *n'*

assumes $\langle c \#_{n'} \text{inf-llist } t \rangle \geq \langle c \#_n \text{inf-llist } t \rangle$

shows $n' \geq \langle c \Leftarrow t \rangle_n$

proof *cases*

assume $\langle c \#_{n'} \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$

thus *?thesis*

proof *cases*

assume $n' \geq n$

thus *?thesis*

by (*rule order-trans*[*OF lNActLe*])

next

assume $\neg n' \geq n$

hence $n' < n$ **by** *simp*

with $\langle c \#_{n'} \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$ **have** $\nexists k. k \geq n' \wedge k < n \wedge \|c\|_t k$

by (*metis enat-ord-simps(1) enat-ord-simps(2) nAct-same-not-active*)

thus *?thesis* **using** *lNActLe-nact* **by** (*simp add: <n' < n>*)

qed

next

assume $\neg \langle c \#_{n'} \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$

with *assms* **have** $\langle c \#_{\text{enat } n'} \text{inf-llist } t \rangle > \langle c \#_{\text{enat } n} \text{inf-llist } t \rangle$ **by** *simp*

hence $n' > n$ **using** *nAct-strict-mono-back*[of *c enat n inf-llist t enat n'*] **by** *simp*

thus *?thesis* **by** (*meson dual-order.strict-implies-order lNActLe le-trans*)

qed

lemma *nAct-mono-lNAct*:

assumes $\langle c \Leftarrow t \rangle_n \leq n'$

shows $\langle c \#_n \text{inf-llist } t \rangle \leq \langle c \#_{n'} \text{inf-llist } t \rangle$

proof –

have $\nexists k. k \geq \langle c \Leftarrow t \rangle_n \wedge k < n \wedge \|c\|_t k$ **using** *lNAct-notActive* **by** *auto*

moreover **have** $\text{enat } n - 1 < \text{llength } (\text{inf-llist } t)$ **by** (*simp add: one-enat-def*)

moreover **from** $\langle c \Leftarrow t \rangle_n \leq n'$ **have** $\text{enat } \langle c \Leftarrow t \rangle_n \leq \text{enat } n$ **by** *simp*

ultimately **have** $\langle c \#_n \text{inf-llist } t \rangle = \langle c \#_{\langle c \Leftarrow t \rangle_n} \text{inf-llist } t \rangle$ **using** *nAct-not-active-same* **by** *simp*

thus *?thesis* **using** *nAct-mono assms* **by** *simp*

qed

1.10 Next Active

In the following, we introduce an operator to obtain the next point in time when a component is activated.

definition *nextAct* :: '*id* \Rightarrow (*nat* \Rightarrow *cnf*) \Rightarrow *nat* \Rightarrow *nat* ($\langle - \rightarrow - \rangle$)

where $\langle c \rightarrow t \rangle_n \equiv (\text{THE } n'. n' \geq n \wedge \|c\|_t n' \wedge (\nexists k. k \geq n \wedge k < n' \wedge \|c\|_t k))$

lemma *nextActI*:

fixes *n::nat*

and *t::nat* \Rightarrow *cnf*

and *c::'id*

assumes $\exists i \geq n. \|c\|_t i$
shows $\langle c \rightarrow t \rangle_n \geq n \wedge \|c\|_t \langle c \rightarrow t \rangle_n \wedge (\nexists k. k \geq n \wedge k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k)$
proof –
let $?P = THE\ n'.\ n' \geq n \wedge \|c\|_t n' \wedge (\nexists k. k \geq n \wedge k < n' \wedge \|c\|_t k)$
from *assms* **obtain** i **where** $i \geq n \wedge \|c\|_t i \wedge (\nexists k. k \geq n \wedge k < i \wedge \|c\|_t k)$
using *lActive-least*[of n *inf-llist* $t\ c$] **by** *auto*
moreover **have** $(\wedge x. n \leq x \wedge \|c\|_t x \wedge \neg (\exists k \geq n. k < x \wedge \|c\|_t k) \implies x = i)$
proof –
fix x **assume** $n \leq x \wedge \|c\|_t x \wedge \neg (\exists k \geq n. k < x \wedge \|c\|_t k)$
show $x = i$
proof (*rule ccontr*)
assume $\neg (x = i)$
thus *False* **using** $\langle i \geq n \wedge \|c\|_t i \wedge (\nexists k. k \geq n \wedge k < i \wedge \|c\|_t k) \rangle$
 $\langle n \leq x \wedge \|c\|_t x \wedge \neg (\exists k \geq n. k < x \wedge \|c\|_t k) \rangle$ **by** *fastforce*
qed
qed
ultimately **have** $(?P) \geq n \wedge \|c\|_t (?P) \wedge (\nexists k. k \geq n \wedge k < ?P \wedge \|c\|_t k)$
using *theI*[of $\lambda n'. n' \geq n \wedge \|c\|_t n' \wedge (\nexists k. k \geq n \wedge k < n' \wedge \|c\|_t k)$] **by** *blast*
thus *?thesis* **using** *nxtAct-def*[of $c\ t\ n$] **by** *metis*
qed

lemma *nxtActLe*:
fixes $n\ n'$
assumes $\exists i \geq n. \|c\|_t i$
shows $n \leq \langle c \rightarrow t \rangle_n$
by (*simp add: assms nxtActI*)

lemma *nxtAct-eq*:
assumes $n' \geq n$
and $\|c\|_t n'$
and $\forall n'' \geq n. n'' < n' \longrightarrow \neg \|c\|_t n''$
shows $n' = \langle c \rightarrow t \rangle_n$
by (*metis assms(1) assms(2) assms(3) nxtActI linorder-neqE-nat nxtActLe*)

lemma *nxtAct-active*:
fixes $i::nat$
and $t::nat \Rightarrow cnf$
and $c::'id$
assumes $\|c\|_t i$
shows $\langle c \rightarrow t \rangle_i = i$ **by** (*metis assms le-eq-less-or-eq nxtActI*)

lemma *nxtActive-no-active*:
assumes $\exists! i. i \geq n \wedge \|c\|_t i$
shows $\neg (\exists i' \geq Suc\ \langle c \rightarrow t \rangle_n. \|c\|_t i')$

proof
assume $\exists i' \geq Suc\ \langle c \rightarrow t \rangle_n. \|c\|_t i'$
then **obtain** i' **where** $i' \geq Suc\ \langle c \rightarrow t \rangle_n$ **and** $\|c\|_t i'$ **by** *auto*
moreover **from** *assms(1)* **have** $\langle c \rightarrow t \rangle_n \geq n$ **using** *nxtActI* **by** *auto*
ultimately **have** $i' \geq n$ **and** $\|c\|_t i'$ **and** $i' \neq \langle c \rightarrow t \rangle_n$ **by** *auto*
moreover **from** *assms(1)* **have** $\|c\|_t \langle c \rightarrow t \rangle_n$ **and** $\langle c \rightarrow t \rangle_n \geq n$ **using** *nxtActI* **by** *auto*
ultimately **show** *False* **using** *assms(1)* **by** *auto*
qed

lemma *nxt-geq-lNact*[*simp*]:
assumes $\exists i \geq n. \|c\|_t i$

shows $\langle c \rightarrow t \rangle_n \geq \langle c \Leftarrow t \rangle_n$
proof –
 from *assms* have $n \leq \langle c \rightarrow t \rangle_n$ **using** *nxtActLe* **by** *simp*
 moreover have $\langle c \Leftarrow t \rangle_n \leq n$ **by** *simp*
 ultimately show *?thesis* **by** *arith*
qed

lemma *active-geq-nxtAct*:

assumes $\|c\|_t i$
 and *the-enat* ($\langle c \#_i \text{inf-llist } t \rangle \geq \text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)$)
 shows $i \geq \langle c \rightarrow t \rangle_n$

proof *cases*

assume $\langle c \#_i \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$

show *?thesis*

proof (*rule ccontr*)

assume $\neg i \geq \langle c \rightarrow t \rangle_n$

hence $i < \langle c \rightarrow t \rangle_n$ **by** *simp*

with $\langle c \#_i \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$ **have** $\neg (\exists k \geq i. k < n \wedge \|c\|_t k)$

by (*metis enat-ord-simps(1) leD leI nAct-same-not-active*)

moreover have $\neg (\exists k \geq n. k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k)$ **using** *nxtActI* **by** *blast*

ultimately have $\neg (\exists k \geq i. k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k)$ **by** *auto*

with $\langle i < \langle c \rightarrow t \rangle_n \rangle$ **show** *False* **using** $\langle \|c\|_t i \rangle$ **by** *simp*

qed

next

assume $\neg \langle c \#_i \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$

moreover from $\langle \text{the-enat } (\langle c \#_i \text{inf-llist } t \rangle) \geq \text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle) \rangle$

have $\langle c \#_i \text{inf-llist } t \rangle \geq \langle c \#_n \text{inf-llist } t \rangle$

by (*metis enat.distinct(2) enat-ord-simps(1) nAct-enat-the-nat*)

ultimately have $\langle c \#_i \text{inf-llist } t \rangle > \langle c \#_n \text{inf-llist } t \rangle$ **by** *simp*

hence $i > n$ **using** *nAct-strict-mono-back[of c n inf-llist t i]* **by** *simp*

with $\langle \|c\|_t i \rangle$ **show** *?thesis* **by** (*meson dual-order.strict-implies-order leI nxtActI*)

qed

lemma *nAct-same*:

assumes $\langle c \Leftarrow t \rangle_n \leq n'$ and $n' \leq \langle c \rightarrow t \rangle_n$

shows *the-enat* ($\langle c \#_{\text{enat } n'} \text{inf-llist } t \rangle = \text{the-enat } (\langle c \#_{\text{enat } n} \text{inf-llist } t \rangle)$)

proof *cases*

assume $n \leq n'$

moreover have $n' - 1 < \text{llength } (\text{inf-llist } t)$ **by** *simp*

moreover have $\neg (\exists i \geq n. i < n' \wedge \|c\|_t i)$ **by** (*meson assms(2) less-le-trans nxtActI*)

ultimately show *?thesis* **using** *nAct-not-active-same* **by** (*simp add: one-enat-def*)

next

assume $\neg n \leq n'$

hence $n' < n$ **by** *simp*

moreover have $n - 1 < \text{llength } (\text{inf-llist } t)$ **by** *simp*

moreover have $\neg (\exists i \geq n'. i < n \wedge \|c\|_t i)$ **by** (*metis* $\langle \neg n \leq n' \rangle$ *assms(1) dual-order.trans lNAct-ex*)

ultimately show *?thesis* **using** *nAct-not-active-same[of n' n]* **by** (*simp add: one-enat-def*)

qed

lemma *nAct-mono-nxtAct*:

assumes $\exists i \geq n. \|c\|_t i$

and $\langle c \rightarrow t \rangle_n \leq n'$

shows $\langle c \#_n \text{inf-llist } t \rangle \leq \langle c \#_{n'} \text{inf-llist } t \rangle$

proof –

from *assms* have $\langle c \#_{\langle c \rightarrow t \rangle_n} \text{inf-llist } t \rangle \leq \langle c \#_{n'} \text{inf-llist } t \rangle$ **using** *nAct-mono assms* **by** *simp*

moreover have $\langle c \#_{\langle c \rightarrow t \rangle_n} \text{inf-llist } t \rangle = \langle c \#_n \text{inf-llist } t \rangle$
proof –
from *assms* **have** $\nexists k. k \geq n \wedge k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k$ **and** $n \leq \langle c \rightarrow t \rangle_n$ **using** *nextActI* **by** *auto*
moreover have *enat* $\langle c \rightarrow t \rangle_n - 1 < \text{length}(\text{inf-llist } t)$ **by** (*simp add: one-enat-def*)
ultimately show *?thesis* **using** *nAct-not-active-same*[of $n \langle c \rightarrow t \rangle_n$] **by** *auto*
qed
ultimately show *?thesis* **by** *simp*
qed

1.11 Latest Activation

In the following, we introduce an operator to obtain the latest point in time when a component is activated.

abbreviation *latestAct-cond*:: $'id \Rightarrow \text{trace} \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow \text{bool}$
where *latestAct-cond* $c \ t \ n \ n' \equiv n' < n \wedge \|c\|_t n'$

definition *latestAct*:: $'id \Rightarrow \text{trace} \Rightarrow \text{nat} \Rightarrow \text{nat} (\langle - \leftarrow - \rangle)$
where *latestAct* $c \ t \ n = (\text{GREATEST } n'. \text{latestAct-cond } c \ t \ n \ n')$

lemma *latestActEx*:

assumes $\exists n' < n. \|nid\|_t n'$
shows $\exists n'. \text{latestAct-cond } nid \ t \ n \ n' \wedge (\forall n''. \text{latestAct-cond } nid \ t \ n \ n'' \longrightarrow n'' \leq n')$
proof –
from *assms* **obtain** n' **where** *latestAct-cond* $nid \ t \ n \ n'$ **by** *auto*
moreover have $\forall n'' > n. \neg \text{latestAct-cond } nid \ t \ n \ n''$ **by** *simp*
ultimately obtain n' **where** *latestAct-cond* $nid \ t \ n \ n' \wedge (\forall n''. \text{latestAct-cond } nid \ t \ n \ n'' \longrightarrow n'' \leq n')$
using *boundedGreatest*[of *latestAct-cond* $nid \ t \ n \ n'$] **by** *blast*
thus *?thesis* ..
qed

lemma *latestAct-prop*:

assumes $\exists n' < n. \|nid\|_t n'$
shows $\|nid\|_t (\text{latestAct } nid \ t \ n)$ **and** *latestAct* $nid \ t \ n < n$
proof –
from *assms* *latestActEx* **have** *latestAct-cond* $nid \ t \ n (\text{GREATEST } x. \text{latestAct-cond } nid \ t \ n \ x)$
using *GreatestI-ex-nat*[of *latestAct-cond* $nid \ t \ n$] **by** *blast*
thus $\|nid\|_t \langle nid \leftarrow t \rangle_n$ **and** *latestAct* $nid \ t \ n < n$ **using** *latestAct-def* **by** *auto*
qed

lemma *latestAct-less*:

assumes *latestAct-cond* $nid \ t \ n \ n'$
shows $n' \leq \langle nid \leftarrow t \rangle_n$
proof –
from *assms* *latestActEx* **have** $n' \leq (\text{GREATEST } x. \text{latestAct-cond } nid \ t \ n \ x)$
using *Greatest-le-nat*[of *latestAct-cond* $nid \ t \ n$] **by** *blast*
thus *?thesis* **using** *latestAct-def* **by** *auto*
qed

lemma *latestActNxt*:

assumes $\exists n' < n. \|nid\|_t n'$
shows $\langle nid \rightarrow t \rangle_{\langle nid \leftarrow t \rangle_n} = \langle nid \leftarrow t \rangle_n$
using *assms* *latestAct-prop*(1) *nextAct-active* **by** *auto*

lemma *latestActNxtAct*:
assumes $\exists n' \geq n. \|tid\|_t n'$
and $\exists n' < n. \|tid\|_t n'$
shows $\langle tid \rightarrow t \rangle_n > \langle tid \leftarrow t \rangle_n$
by (*meson assms latestAct-prop(2) less-le-trans nxtActI zero-le*)

lemma *latestActless*:
assumes $\exists n' \geq n_s. n' < n \wedge \|nid\|_t n'$
shows $\langle nid \leftarrow t \rangle_{n \geq n_s}$
by (*meson assms dual-order.trans latestAct-less*)

lemma *latestActEq*:
fixes *nid::'id*
assumes $\|nid\|_t n'$ **and** $\neg(\exists n'' > n'. n'' < n \wedge \|nid\|_t n'')$ **and** $n' < n$
shows $\langle nid \leftarrow t \rangle_n = n'$
using *latestAct-def*

proof
have (*GREATEST* $n'. \text{latestAct-cond } nid \ t \ n \ n'$) = n'
proof (*rule Greatest-equality[of latestAct-cond nid t n n']*)
from *assms(1) assms(3)* **show** *latestAct-cond nid t n n'* **by** *simp*
next
fix *y* **assume** *latestAct-cond nid t n y*
hence $\|nid\|_t y$ **and** $y < n$ **by** *auto*
thus $y \leq n'$ **using** *assms(1) assms(2) leI* **by** *blast*
qed
thus $n' = (\text{GREATEST } n'. \text{latestAct-cond } nid \ t \ n \ n')$ **by** *simp*
qed

1.12 Last Activation

In the following we introduce an operator to obtain the latest point in time where a certain component was activated within a certain configuration trace.

definition *lActive* :: *'id* \Rightarrow (*nat* \Rightarrow *cnf*) \Rightarrow *nat* ($\langle - \wedge - \rangle$)
where $\langle c \wedge t \rangle \equiv (\text{GREATEST } i. \|c\|_t i)$

lemma *lActive-active*:
assumes $\|c\|_t i$
and $\forall n' > n. \neg(\|c\|_t n')$
shows $\|c\|_t (\langle c \wedge t \rangle)$
proof –
from *assms* **obtain** *i'* **where** $\|c\|_t i'$ **and** $\bigwedge y. \|c\|_t y \implies y \leq i'$
using *boundedGreatest[of $\lambda i'. \|c\|_t i', i \ n$]* **by** *blast*
thus *?thesis* **using** *lActive-def Nat.GreatestI-nat[of $\lambda i'. \|c\|_t i'$]* **by** *metis*
qed

lemma *lActive-less*:
assumes $\|c\|_t i$
and $\forall n' > n. \neg(\|c\|_t n')$
shows $\langle c \wedge t \rangle \leq n$
proof (*rule ccontr*)
assume $\neg \langle c \wedge t \rangle \leq n$
hence $\langle c \wedge t \rangle > n$ **by** *simp*
moreover from *assms* **have** $\|c\|_t (\langle c \wedge t \rangle)$ **using** *lActive-active* **by** *simp*
ultimately show *False* **using** *assms* **by** *simp*
qed

lemma *lActive-greatest:*

assumes $\|c\|_t i$
and $\forall n' > n. \neg (\|c\|_t n')$
shows $i \leq \langle c \wedge t \rangle$

proof –

from *assms* **obtain** i' **where** $\|c\|_t i'$ **and** $\bigwedge y. \|c\|_t y \implies y \leq i'$
using *boundedGreatest*[of $\lambda i'. \|c\|_t i', i n$] **by** *blast*
with *assms* **show** *?thesis* **using** *lActive-def Nat.Greatest-le-nat*[of $\lambda i'. \|c\|_t i', i$] **by** *metis*
qed

lemma *lActive-greater-active:*

assumes $n > \langle c \wedge t \rangle$
and $\forall n'' > n'. \neg \|c\|_t n''$
shows $\neg \|c\|_t n$

proof (*rule ccontr*)

assume $\neg \neg \|c\|_t n$
with $\langle \forall n'' > n'. \neg \|c\|_t n'' \rangle$ **have** $n \leq \langle c \wedge t \rangle$ **using** *lActive-greatest* **by** *simp*
thus *False* **using** *assms* **by** *simp*

qed

lemma *lActive-greater-active-all:*

assumes $\forall n'' > n'. \neg \|c\|_t n''$
shows $\neg (\exists n > \langle c \wedge t \rangle. \|c\|_t n)$

proof (*rule ccontr*)

assume $\neg \neg (\exists n > \langle c \wedge t \rangle. \|c\|_t n)$
then obtain n **where** $n > \langle c \wedge t \rangle$ **and** $\|c\|_t n$ **by** *blast*
with $\langle \forall n'' > n'. \neg (\|c\|_t n'') \rangle$ **have** $\neg \|c\|_t n$ **using** *lActive-greater-active* **by** *simp*
with $\langle \|c\|_t n \rangle$ **show** *False* **by** *simp*

qed

lemma *lActive-equality:*

assumes $\|c\|_t i$
and $(\bigwedge x. \|c\|_t x \implies x \leq i)$
shows $\langle c \wedge t \rangle = i$ **unfolding** *lActive-def* **using** *assms Greatest-equality*[of $\lambda i'. \|c\|_t i'$] **by** *simp*

lemma *nxtActive-lactive:*

assumes $\exists i \geq n. \|c\|_t i$
and $\neg (\exists i > \langle c \rightarrow t \rangle_n. \|c\|_t i)$
shows $\langle c \rightarrow t \rangle_n = \langle c \wedge t \rangle$

proof –

from *assms*(1) **have** $\|c\|_t \langle c \rightarrow t \rangle_n$ **using** *nxtActI* **by** *auto*
moreover from *assms* **have** $\neg (\exists i' \geq \text{Suc } \langle c \rightarrow t \rangle_n. \|c\|_t i')$ **using** *nxtActive-no-active* **by** *simp*
hence $(\bigwedge x. \|c\|_t x \implies x \leq \langle c \rightarrow t \rangle_n)$ **using** *not-less-eq-eq* **by** *auto*
ultimately show *?thesis* **using** $\langle \neg (\exists i' \geq \text{Suc } \langle c \rightarrow t \rangle_n. \|c\|_t i') \rangle$ *lActive-equality* **by** *simp*

qed

1.13 Mapping Time Points

In the following we introduce two operators to map time-points between configuration traces and behavior traces.

1.13.1 Configuration Trace to Behavior Trace

First we provide an operator which maps a point in time of a configuration trace to the corresponding point in time of a behavior trace.

definition $cnf2bhv :: 'id \Rightarrow (nat \Rightarrow cnf) \Rightarrow nat \Rightarrow nat (\cdot \downarrow \cdot) [150,150,150] 110)$
where $c \downarrow_t(n) \equiv the-enat(llength (\pi_c(inf-llist t))) - 1 + (n - \langle c \wedge t \rangle)$

lemma $cnf2bhv-mono$:
assumes $n' \geq n$
shows $c \downarrow_t(n') \geq c \downarrow_t(n)$
by (*simp add: assms cnf2bhv-def diff-le-mono*)

lemma $cnf2bhv-mono-strict$:
assumes $n \geq \langle c \wedge t \rangle$ **and** $n' > n$
shows $c \downarrow_t(n') > c \downarrow_t(n)$
using *assms cnf2bhv-def by auto*

Note that the functions are nat, that means that also in the case the difference is negative they will return a 0!

lemma $cnf2bhv-ge-llength[*simp*]$:
assumes $n \geq \langle c \wedge t \rangle$
shows $c \downarrow_t(n) \geq the-enat(llength (\pi_c(inf-llist t))) - 1$
using *assms cnf2bhv-def by simp*

lemma $cnf2bhv-greater-llength[*simp*]$:
assumes $n > \langle c \wedge t \rangle$
shows $c \downarrow_t(n) > the-enat(llength (\pi_c(inf-llist t))) - 1$
using *assms cnf2bhv-def by simp*

lemma $cnf2bhv-suc[*simp*]$:
assumes $n \geq \langle c \wedge t \rangle$
shows $c \downarrow_t(Suc n) = Suc (c \downarrow_t(n))$
using *assms cnf2bhv-def by simp*

lemma $cnf2bhv-lActive[*simp*]$:
shows $c \downarrow_t(\langle c \wedge t \rangle) = the-enat(llength (\pi_c(inf-llist t))) - 1$
using *cnf2bhv-def by simp*

lemma $cnf2bhv-lnth-lappend$:
assumes $act: \exists i. \|c\|_t i$
and $nAct: \nexists i. i \geq n \wedge \|c\|_t i$
shows $lnth ((\pi_c(inf-llist t)) @_l (inf-llist t')) (c \downarrow_t(n)) = lnth (inf-llist t') (n - \langle c \wedge t \rangle - 1)$
(is ?lhs = ?rhs)

proof –

from $nAct$ **have** $lfinite (\pi_c(inf-llist t))$ **using** *proj-finite2 by auto*
then obtain k **where** $k-def: llength (\pi_c(inf-llist t)) = enat k$ **using** *lfinite-llength-enat by blast*
moreover have $k \leq c \downarrow_t(n)$

proof –

from $nAct$ **have** $\nexists i. i > n - 1 \wedge \|c\|_t i$ **by** *simp*
with act **have** $\langle c \wedge t \rangle \leq n - 1$ **using** *lActive-less by auto*
moreover have $n > 0$ **using** *act nAct by auto*
ultimately have $\langle c \wedge t \rangle < n$ **by** *simp*
hence $the-enat (llength (\pi_c(inf-llist t))) - 1 < c \downarrow_t(n)$ **using** *cnf2bhv-greater-llength by simp*
with $k-def$ **show** *?thesis* **by** *simp*

qed

ultimately have $?lhs = lnth (inf-llist t') (c \downarrow_t(n) - k)$ **using** $lnth-lappend2$ **by** $blast$
 moreover have $c \downarrow_t(n) - k = n - \langle c \wedge t \rangle - 1$

proof –

from $cnf2bhv-def$ **have** $c \downarrow_t(n) - k = the-enat (llength (\pi_c inf-llist t)) - 1 + (n - \langle c \wedge t \rangle) - k$
by $simp$

also have $\dots = the-enat (llength (\pi_c inf-llist t)) - 1 + (n - \langle c \wedge t \rangle) -$
 $the-enat (llength (\pi_c (inf-llist t)))$ **using** $k-def$ **by** $simp$

also have $\dots = the-enat (llength (\pi_c inf-llist t)) + (n - \langle c \wedge t \rangle) - 1 -$
 $the-enat (llength (\pi_c (inf-llist t)))$

proof –

have $\exists i. enat i < llength (inf-llist t) \wedge \|c\|_{lnth (inf-llist t) i}$ **by** $(simp\ add: act)$

hence $llength (\pi_c inf-llist t) \geq 1$ **using** $proj-one$ **by** $simp$

moreover from $k-def$ **have** $llength (\pi_c inf-llist t) \neq \infty$ **by** $simp$

ultimately have $the-enat (llength (\pi_c inf-llist t)) \geq 1$ **by** $(simp\ add: k-def\ one-enat-def)$

thus $?thesis$ **by** $simp$

qed

also have $\dots = the-enat (llength (\pi_c inf-llist t)) + (n - \langle c \wedge t \rangle) -$
 $the-enat (llength (\pi_c (inf-llist t))) - 1$ **by** $simp$

also have $\dots = n - \langle c \wedge t \rangle - 1$ **by** $simp$

finally show $?thesis$.

qed

ultimately show $?thesis$ **by** $simp$

qed

lemma $nAct-cnfg2proj-Suc-dist$:

assumes $\exists i \geq n. \|c\|_t i$

and $\neg(\exists i > \langle c \rightarrow t \rangle_n. \|c\|_t i)$

shows $Suc (the-enat \langle c \#_{enat} n inf-llist t \rangle) = c \downarrow_t (Suc \langle c \rightarrow t \rangle_n)$

proof –

have $the-enat \langle c \#_{enat} n inf-llist t \rangle = c \downarrow_t (\langle c \rightarrow t \rangle_n)$ **(is** $?LHS = ?RHS)$

proof –

from $assms$ **have** $?RHS = the-enat (llength (\pi_c (inf-llist t))) - 1$

using $nxtActive-lactive[of\ n\ c\ t]$ **by** $simp$

also have $llength (\pi_c (inf-llist t)) = eSuc (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle)$

proof –

from $assms$ **have** $\neg(\exists i' \geq Suc (\langle c \rightarrow t \rangle_n). \|c\|_t i')$ **using** $nxtActive-no-active$ **by** $simp$

hence $\langle c \#_{Suc (\langle c \rightarrow t \rangle_n)} inf-llist t \rangle = llength (\pi_c (inf-llist t))$

using $nAct-eq-proj[of\ Suc (\langle c \rightarrow t \rangle_n)\ c\ inf-llist\ t]$ **by** $simp$

moreover from $assms(1)$ **have** $\|c\|_t (\langle c \rightarrow t \rangle_n)$ **using** $nxtActI$ **by** $blast$

hence $\langle c \#_{Suc (\langle c \rightarrow t \rangle_n)} inf-llist t \rangle = eSuc (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle)$ **by** $simp$

ultimately show $?thesis$ **by** $simp$

qed

also have $the-enat (eSuc (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle)) - 1 = (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle)$

proof –

have $\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle \neq \infty$ **by** $simp$

hence $the-enat (eSuc (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle)) = Suc (the-enat (\langle c \#_{\langle c \rightarrow t \rangle_n} inf-llist t \rangle))$

using $the-enat-eSuc$ **by** $simp$

thus $?thesis$ **by** $simp$

qed

also have $\dots = ?LHS$

proof –

have $enat \langle c \rightarrow t \rangle_n - 1 < llength (inf-llist t)$ **by** $(simp\ add: one-enat-def)$

moreover from $assms(1)$ **have** $\langle c \rightarrow t \rangle_n \geq n$ **and**

$\nexists k. enat n \leq enat k \wedge enat k < enat \langle c \rightarrow t \rangle_n \wedge \|c\|_{lnth (inf-llist t) k}$ **using** $nxtActI$ **by** $auto$

ultimately have $\langle c \#_{\text{enat}} \langle c \rightarrow t \rangle_n \text{inf-llist } t \rangle = \langle c \#_{\text{enat}} n \text{inf-llist } t \rangle$
using $n \text{Act-not-active-same}[of\ n\ \langle c \rightarrow t \rangle_n \text{inf-llist } t\ c]$ **by** *simp*
moreover have $\langle c \#_{\text{enat}} n \text{inf-llist } t \rangle \neq \infty$ **by** *simp*
ultimately show *?thesis* **by** *auto*
qed
finally show *?thesis* **by** *fastforce*
qed
moreover from *assms* **have** $\langle c \rightarrow t \rangle_n = \langle c \wedge t \rangle$ **using** *nextActive-lactive* **by** *simp*
hence $\text{Suc } (c \downarrow_t (\langle c \rightarrow t \rangle_n)) = c \downarrow_t (\text{Suc } \langle c \rightarrow t \rangle_n)$ **using** *cnf2bhv-suc* [**where** $n = \langle c \rightarrow t \rangle_n$] **by** *simp*
ultimately show *?thesis* **by** *simp*
qed

1.13.2 Behavior Trace to Configuration Trace

Next we define an operator to map a point in time of a behavior trace back to a corresponding point in time for a configuration trace.

definition $\text{bhv2cnf} :: 'id \Rightarrow (\text{nat} \Rightarrow \text{cnf}) \Rightarrow \text{nat} \Rightarrow \text{nat} \Rightarrow (_ \uparrow _ -) [150, 150, 150] 110)$
where $c \uparrow_t (n) \equiv \langle c \wedge t \rangle + (n - (\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t)))) - 1)$

lemma *bhv2cnf-mono*:
assumes $n' \geq n$
shows $c \uparrow_t (n') \geq c \uparrow_t (n)$
by (*simp add: assms bhv2cnf-def diff-le-mono*)

lemma *bhv2cnf-mono-strict*:
assumes $n' > n$
and $n \geq \text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) - 1$
shows $c \uparrow_t (n') > c \uparrow_t (n)$
using *assms bhv2cnf-def* **by** *auto*

Note that the functions are nat, that means that also in the case the difference is negative they will return a 0!

lemma *bhv2cnf-ge-lActive*[*simp*]:
shows $c \uparrow_t (n) \geq \langle c \wedge t \rangle$
using *bhv2cnf-def* **by** *simp*

lemma *bhv2cnf-greater-lActive*[*simp*]:
assumes $n > \text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) - 1$
shows $c \uparrow_t (n) > \langle c \wedge t \rangle$
using *assms bhv2cnf-def* **by** *simp*

lemma *bhv2cnf-lActive*[*simp*]:
assumes $\exists i. \parallel c \parallel_t i$
and $\text{lfinite } (\pi_c(\text{inf-llist } t))$
shows $c \uparrow_t (\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t)))) = \text{Suc } (\langle c \wedge t \rangle)$

proof –

from *assms* **have** $\pi_c(\text{inf-llist } t) \neq []_t$ **by** *simp*
hence $\text{llength } (\pi_c(\text{inf-llist } t)) > 0$ **by** (*simp add: lnull-def*)
moreover from $\langle \text{lfinite } (\pi_c(\text{inf-llist } t)) \rangle$ **have** $\text{llength } (\pi_c(\text{inf-llist } t)) \neq \infty$
using *llength-eq-infty-conv-lfinite* **by** *auto*
ultimately have $\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) > 0$ **using** *enat-0-iff(1)* **by** *fastforce*
hence $\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) - (\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t)))) - 1 = 1$ **by** *simp*
thus *?thesis* **using** *bhv2cnf-def* **by** *simp*

qed

1.13.3 Relating the Mappings

In the following we provide some properties about the relationship between the two mapping operators.

lemma *bhv2cnf-cnf2bhv*:

assumes $n \geq \langle c \wedge t \rangle$

shows $c \uparrow_t(c \downarrow_t(n)) = n$ (**is** *?lhs = ?rhs*)

proof –

have $?lhs = \langle c \wedge t \rangle + ((c \downarrow_t(n)) - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1)$

using *bhv2cnf-def* **by** *simp*

also have $\dots = \langle c \wedge t \rangle + (((the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1 + (n - \langle c \wedge t \rangle)) - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1)$ **using** *cnf2bhv-def* **by** *simp*

also have $(the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1 + (n - \langle c \wedge t \rangle) - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1 = (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1 - ((the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1) + (n - \langle c \wedge t \rangle)$ **by** *simp*

also have $\dots = n - \langle c \wedge t \rangle$ **by** *simp*

also have $\langle c \wedge t \rangle + (n - \langle c \wedge t \rangle) = \langle c \wedge t \rangle + n - \langle c \wedge t \rangle$ **using** *assms* **by** *simp*

also have $\dots = ?rhs$ **by** *simp*

finally show *?thesis* .

qed

lemma *cnf2bhv-bhv2cnf*:

assumes $n \geq the-enat(\text{length}(\pi_c(\text{inf-llist } t))) - 1$

shows $c \downarrow_t(c \uparrow_t(n)) = n$ (**is** *?lhs = ?rhs*)

proof –

have $?lhs = the-enat(\text{length}(\pi_c(\text{inf-llist } t))) - 1 + ((c \uparrow_t(n)) - \langle c \wedge t \rangle)$

using *cnf2bhv-def* **by** *simp*

also have $\dots = the-enat(\text{length}(\pi_c(\text{inf-llist } t))) - 1 + \langle c \wedge t \rangle + (n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1) - \langle c \wedge t \rangle$ **using** *bhv2cnf-def* **by** *simp*

also have $\langle c \wedge t \rangle + (n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1) - \langle c \wedge t \rangle = \langle c \wedge t \rangle - \langle c \wedge t \rangle + (n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1)$ **by** *simp*

also have $\dots = n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1$ **by** *simp*

also have $the-enat(\text{length}(\pi_c(\text{inf-llist } t))) - 1 + (n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1) = n - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1 + (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1$ **by** *simp*

also have $\dots = n + ((the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1) - (the-enat(\text{length}(\pi_c(\text{inf-llist } t)))) - 1$ **using** *assms* **by** *simp*

also have $\dots = ?rhs$ **by** *simp*

finally show *?thesis* .

qed

lemma *p2c-mono-c2p*:

assumes $n \geq \langle c \wedge t \rangle$

and $n' \geq c \downarrow_t(n)$

shows $c \uparrow_t(n') \geq n$

proof –

from $\langle n' \geq c \downarrow_t(n) \rangle$ **have** $c \uparrow_t(n') \geq c \uparrow_t(c \downarrow_t(n))$ **using** *bhv2cnf-mono* **by** *simp*

thus *?thesis* **using** *bhv2cnf-cnf2bhv* $\langle n \geq \langle c \wedge t \rangle \rangle$ **by** *simp*

qed

lemma *p2c-mono-c2p-strict*:

assumes $n \geq \langle c \wedge t \rangle$

and $n < c \uparrow_t(n')$

shows $c \downarrow_t(n) < n'$

proof (*rule ccontr*)

assume $\neg (c \downarrow_t(n) < n')$

hence $c\downarrow_t(n) \geq n'$ **by** *simp*
with $\langle n \geq \langle c \wedge t \rangle \rangle$ **have** $c\uparrow_t(\text{nat}(c\downarrow_t(n))) \geq c\uparrow_t(n')$
using *bhv2cnf-mono* **by** *simp*
hence $\neg(c\uparrow_t(\text{nat}(c\downarrow_t(n))) < c\uparrow_t(n'))$ **by** *simp*
with $\langle n \geq \langle c \wedge t \rangle \rangle$ **have** $\neg(n < c\uparrow_t(n'))$
using *bhv2cnf-cnf2bhv* **by** *simp*
with *assms* **show** *False* **by** *simp*
qed

lemma *c2p-mono-p2c*:

assumes $n \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1$
and $n' \geq c\uparrow_t(n)$
shows $c\downarrow_t(n') \geq n$

proof –

from $\langle n' \geq c\uparrow_t(n) \rangle$ **have** $c\downarrow_t(n') \geq c\downarrow_t(c\uparrow_t(n))$ **using** *cnf2bhv-mono* **by** *simp*
thus *?thesis* **using** *cnf2bhv-bhv2cnf* $\langle n \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1 \rangle$ **by** *simp*

qed

lemma *c2p-mono-p2c-strict*:

assumes $n \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1$
and $n < c\downarrow_t(n')$
shows $c\uparrow_t(n) < n'$

proof (*rule ccontr*)

assume $\neg(c\uparrow_t(n) < n')$
hence $c\uparrow_t(n) \geq n'$ **by** *simp*
with $\langle n \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1 \rangle$ **have** $c\downarrow_t(\text{nat}(c\uparrow_t(n))) \geq c\downarrow_t(n')$
using *cnf2bhv-mono* **by** *simp*
hence $\neg(c\downarrow_t(\text{nat}(c\uparrow_t(n))) < c\downarrow_t(n'))$ **by** *simp*
with $\langle n \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1 \rangle$ **have** $\neg(n < c\downarrow_t(n'))$
using *cnf2bhv-bhv2cnf* **by** *simp*
with *assms* **show** *False* **by** *simp*

qed

end

end

2 A Calculus for Dynamic Architectures

The following theory formalizes our calculus for dynamic architectures [2, 3] and verifies its soundness. The calculus allows to reason about temporal-logic specifications of component behavior in a dynamic setting. The theory is based on our theory of configuration traces and introduces the notion of behavior trace assertion to specify component behavior in a dynamic setting.

theory *Dynamic-Architecture-Calculus*

imports *Configuration-Traces*

begin

2.1 Extended Natural Numbers

We first provide one additional property for extended natural numbers.

lemma *the-enat-mono[simp]*:

assumes $m \neq \infty$

and $n \leq m$
shows *the-enat* $n \leq \text{the-enat } m$
using *assms(1)* *assms(2)* *enat-ile* **by** *fastforce*

2.2 Lazy Lists

Finally, we provide an additional property for lazy lists.

lemma *length-geq-enat-lfiniteD*: *length* $xs \leq \text{enat } n \implies \text{lfinite } xs$
using *not-lfinite-length* **by** *force*

context *dynamic-component*
begin

2.3 Dynamic Evaluation of Temporal Operators

In the following we introduce a function to evaluate a behavior trace assertion over a given configuration trace.

type-synonym $'c \text{ bta} = (\text{nat} \Rightarrow 'c) \Rightarrow \text{nat} \Rightarrow \text{bool}$

definition *eval*:: $'id \Rightarrow (\text{nat} \Rightarrow \text{cnf}) \Rightarrow (\text{nat} \Rightarrow 'cmp) \Rightarrow \text{nat}$
 $\Rightarrow 'cmp \text{ bta} \Rightarrow \text{bool}$

where *eval* $cid \ t \ t' \ n \ \gamma \equiv$
 $(\exists i \geq n. \|cid\|_{t \ i}) \wedge \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat } (\langle cid \#_n \text{inf-llist } t \rangle)) \vee$
 $(\exists i. \|cid\|_{t \ i}) \wedge (\nexists i'. i' \geq n \wedge \|cid\|_{t \ i'}) \wedge \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (cid \downarrow_t (n)) \vee$
 $(\nexists i. \|cid\|_{t \ i}) \wedge \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) \ n$

eval takes a component identifier *cid*, a configuration trace *t*, a behavior trace *t'*, and point in time *n* and evaluates behavior trace assertion γ as follows:

- If component *cid* is again activated in the future, γ is evaluated at the next point in time where *cid* is active in *t*.
- If component *cid* is not again activated in the future but it is activated at least once in *t*, then γ is evaluated at the point in time given by $cid \downarrow_t n$.
- If component *cid* is never active in *t*, then γ is evaluated at time point *n*.

The following proposition evaluates definition *eval* by showing that a behavior trace assertion γ holds over configuration trace *t* and continuation *t'* whenever it holds for the concatenation of the corresponding projection with *t'*.

proposition *eval-corr*:

eval $cid \ t \ t' \ 0 \ \gamma \longleftrightarrow \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) \ 0$

proof

assume *eval* $cid \ t \ t' \ 0 \ \gamma$

with *eval-def* **have** $(\exists i \geq 0. \|cid\|_{t \ i}) \wedge$

$\gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) (\text{the-enat } \langle cid \#_{\text{enat}} \text{inf-llist } t \rangle) \vee$

$(\exists i. \|cid\|_{t \ i}) \wedge \neg (\exists i' \geq 0. \|cid\|_{t \ i'}) \wedge \gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) (cid \downarrow_t 0) \vee$

$(\nexists i. \|cid\|_{t \ i}) \wedge \gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) \ 0$ **by** *simp*

thus $\gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) \ 0$

proof

assume $(\exists i \geq 0. \|cid\|_{t \ i}) \wedge \gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) (\text{the-enat } \langle cid \#_{\text{enat}} \text{inf-llist } t \rangle)$

moreover **have** $\text{the-enat } \langle cid \#_{\text{enat}} \text{inf-llist } t \rangle = 0$ **using** *zero-enat-def* **by** *auto*

ultimately show *thesis* **by** *simp*

next
assume $(\exists i. \|cid\|_t i) \wedge \neg (\exists i' \geq 0. \|cid\|_{t, i'}) \wedge \gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) (cid \downarrow t 0) \vee$
 $(\nexists i. \|cid\|_t i) \wedge \gamma (\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')) 0$
thus ?thesis by auto
qed
next
assume $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) 0$
show $\text{eval } cid \ t \ t' \ 0 \ \gamma$
proof cases
assume $\exists i. \|cid\|_t i$
hence $\exists i \geq 0. \|cid\|_{t, i}$ **by simp**
moreover from $\langle \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) 0 \rangle$ **have**
 $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat}(\langle cid \ \#_{\text{enat}} \ 0 \ \text{inf-llist } t \rangle))$
using zero-enat-def by auto
ultimately show ?thesis using eval-def by simp
next
assume $\nexists i. \|cid\|_t i$
with $\langle \gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) 0 \rangle$ **show ?thesis using eval-def by simp**
qed
qed

2.3.1 Simplification Rules

lemma *validCI-act[simp]*:

assumes $\exists i \geq n. \|cid\|_t i$
and $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat}(\langle cid \ \#_n \ \text{inf-llist } t \rangle))$
shows $\text{eval } cid \ t \ t' \ n \ \gamma$
using *assms eval-def by simp*

lemma *validCI-cont[simp]*:

assumes $\exists i. \|cid\|_t i$
and $\nexists i'. i' \geq n \wedge \|cid\|_{t, i'}$
and $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (cid \downarrow t(n))$
shows $\text{eval } cid \ t \ t' \ n \ \gamma$
using *assms eval-def by simp*

lemma *validCI-not-act[simp]*:

assumes $\nexists i. \|cid\|_t i$
and $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) n$
shows $\text{eval } cid \ t \ t' \ n \ \gamma$
using *assms eval-def by simp*

lemma *validCE-act[simp]*:

assumes $\exists i \geq n. \|cid\|_t i$
and $\text{eval } cid \ t \ t' \ n \ \gamma$
shows $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat}(\langle cid \ \#_n \ \text{inf-llist } t \rangle))$
using *assms eval-def by auto*

lemma *validCE-cont[simp]*:

assumes $\exists i. \|cid\|_t i$
and $\nexists i'. i' \geq n \wedge \|cid\|_{t, i'}$
and $\text{eval } cid \ t \ t' \ n \ \gamma$
shows $\gamma (\text{lnth } ((\pi_{cid}(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (cid \downarrow t(n))$
using *assms eval-def by auto*

lemma *validCE-not-act[simp]*:

assumes $\nexists i. \|cid\|_t i$
and $eval\ cid\ t\ t'\ n\ \gamma$
shows $\gamma\ (lnth\ ((\pi_{cid}(inf-llist\ t))\ @_l\ (inf-llist\ t')))\ n$
using *assms eval-def* **by** *auto*

2.3.2 No Activations

proposition *validity1*:

assumes $n \leq n'$
and $\exists i \geq n'. \|c\|_t i$
and $\forall k \geq n. k < n' \longrightarrow \neg \|c\|_t k$
shows $eval\ c\ t\ t'\ n\ \gamma \implies eval\ c\ t\ t'\ n'\ \gamma$

proof –

assume $eval\ c\ t\ t'\ n\ \gamma$
moreover from *assms* **have** $\exists i \geq n. \|c\|_t i$ **by** (*meson order.trans*)
ultimately have $\gamma\ (lnth\ ((\pi_c(inf-llist\ t))\ @_l\ (inf-llist\ t')))\ (the-enat\ (\langle c\ \#_{enat\ n}\ inf-llist\ t \rangle))$
using *validCE-act* **by** *blast*
moreover have $enat\ n' - 1 < llength\ (inf-llist\ t)$ **by** (*simp add: one-enat-def*)
with *assms* **have** $the-enat\ (\langle c\ \#_{enat\ n}\ inf-llist\ t \rangle) = the-enat\ (\langle c\ \#_{enat\ n'}\ inf-llist\ t \rangle)$
using *nAct-not-active-same*[*of n n' inf-llist t c*] **by** *simp*
ultimately have $\gamma\ (lnth\ ((\pi_c(inf-llist\ t))\ @_l\ (inf-llist\ t')))\ (the-enat\ (\langle c\ \#_{enat\ n'}\ inf-llist\ t \rangle))$
by *simp*
with *assms* **show** *?thesis* **using** *validCI-act* **by** *blast*

qed

proposition *validity2*:

assumes $n \leq n'$
and $\exists i \geq n'. \|c\|_t i$
and $\forall k \geq n. k < n' \longrightarrow \neg \|c\|_t k$
shows $eval\ c\ t\ t'\ n'\ \gamma \implies eval\ c\ t\ t'\ n\ \gamma$

proof –

assume $eval\ c\ t\ t'\ n'\ \gamma$
with $\langle \exists i \geq n'. \|c\|_t i \rangle$
have $\gamma\ (lnth\ ((\pi_c(inf-llist\ t))\ @_l\ (inf-llist\ t')))\ (the-enat\ (\langle c\ \#_{enat\ n'}\ inf-llist\ t \rangle))$
using *validCE-act* **by** *blast*
moreover have $enat\ n' - 1 < llength\ (inf-llist\ t)$ **by** (*simp add: one-enat-def*)
with *assms* **have** $the-enat\ (\langle c\ \#_{enat\ n}\ inf-llist\ t \rangle) = the-enat\ (\langle c\ \#_{enat\ n'}\ inf-llist\ t \rangle)$
using *nAct-not-active-same* **by** *simp*
ultimately have $\gamma\ (lnth\ ((\pi_c(inf-llist\ t))\ @_l\ (inf-llist\ t')))\ (the-enat\ (\langle c\ \#_{enat\ n}\ inf-llist\ t \rangle))$
by *simp*
moreover from *assms* **have** $\exists i \geq n. \|c\|_t i$ **by** (*meson order.trans*)
ultimately show *?thesis* **using** *validCI-act* **by** *blast*

qed

2.4 Specification Operators

In the following we introduce some basic operators for behavior trace assertions.

2.4.1 Predicates

Every predicate can be transformed to a behavior trace assertion.

definition $pred :: bool \Rightarrow ('cmp\ bta)$
where $pred\ P \equiv \lambda\ t\ n. P$

lemma *predI*[*intro*]:

```

fixes cid t t' n P
assumes P
shows eval cid t t' n (pred P)
proof cases
  assume  $(\exists i. \|cid\|_t i)$ 
  show ?thesis
  proof cases
    assume  $\exists i \geq n. \|cid\|_t i$ 
    with assms show ?thesis using eval-def pred-def by auto
  next
    assume  $\neg (\exists i \geq n. \|cid\|_t i)$ 
    with assms show ?thesis using eval-def pred-def by auto
  qed
next
  assume  $\neg (\exists i. \|cid\|_t i)$ 
  with assms show ?thesis using eval-def pred-def by auto
qed

```

```

lemma predE[elim]:
  fixes cid t t' n P
  assumes eval cid t t' n (pred P)
  shows P
proof cases
  assume  $(\exists i. \|cid\|_t i)$ 
  show ?thesis
  proof cases
    assume  $\exists i \geq n. \|cid\|_t i$ 
    with assms show ?thesis using eval-def pred-def by auto
  next
    assume  $\neg (\exists i \geq n. \|cid\|_t i)$ 
    with assms show ?thesis using eval-def pred-def by auto
  qed
next
  assume  $\neg (\exists i. \|cid\|_t i)$ 
  with assms show ?thesis using eval-def pred-def by auto
qed

```

2.4.2 True and False

```

abbreviation true :: 'cmp bta
  where true  $\equiv \lambda t n. HOL.True$ 

```

```

abbreviation false :: 'cmp bta
  where false  $\equiv \lambda t n. HOL.False$ 

```

2.4.3 Implication

```

definition imp :: ('cmp bta)  $\Rightarrow$  ('cmp bta)  $\Rightarrow$  ('cmp bta) (infixl  $\longrightarrow^b$  10)
  where  $\gamma \longrightarrow^b \gamma' \equiv \lambda t n. \gamma t n \longrightarrow \gamma' t n$ 

```

```

lemma impI[intro!]:
  assumes eval cid t t' n  $\gamma \longrightarrow eval cid t t' n \gamma'$ 
  shows eval cid t t' n ( $\gamma \longrightarrow^b \gamma'$ )
proof cases
  assume  $\exists i. \|cid\|_t i$ 
  show ?thesis

```


proof cases
assume $\exists i \geq n. \|cid\|_t i$
with $\langle eval\ cid\ t\ t'\ n\ \gamma \longrightarrow eval\ cid\ t\ t'\ n\ \gamma' \rangle$
have $\gamma\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (the\text{-}enat\ \langle cid\ \#_{enat}\ n\ inf\text{-}llist\ t \rangle)$
 $\longrightarrow \gamma'\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (the\text{-}enat\ \langle cid\ \#_{enat}\ n\ inf\text{-}llist\ t \rangle)$
using *eval-def* **by** *blast*
with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *validCI-act* [**where** $\gamma = \lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n$] **by** *blast*
thus *?thesis* **using** *imp-def* **by** *simp*

next
assume $\neg (\exists i \geq n. \|cid\|_t i)$
with $\langle \exists i. \|cid\|_t i \rangle \langle eval\ cid\ t\ t'\ n\ \gamma \longrightarrow eval\ cid\ t\ t'\ n\ \gamma' \rangle$
have $\gamma\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (cid \downarrow t n)$
 $\longrightarrow \gamma'\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (cid \downarrow t n)$ **using** *eval-def* **by** *blast*
with $\langle \exists i. \|cid\|_t i \rangle \langle \neg (\exists i \geq n. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *validCI-cont* [**where** $\gamma = \lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n$] **by** *blast*
thus *?thesis* **using** *imp-def* **by** *simp*

qed

next
assume $\neg (\exists i. \|cid\|_t i)$
with $\langle eval\ cid\ t\ t'\ n\ \gamma \longrightarrow eval\ cid\ t\ t'\ n\ \gamma' \rangle$
have $\gamma\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ n \longrightarrow \gamma'\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ n$
using *eval-def* **by** *blast*
with $\langle \neg (\exists i. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *validCI-not-act* [**where** $\gamma = \lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n$] **by** *blast*
thus *?thesis* **using** *imp-def* **by** *simp*

qed

lemma *impE*[*elim!*]:
assumes $eval\ cid\ t\ t'\ n\ (\gamma \longrightarrow^b \gamma')$
shows $eval\ cid\ t\ t'\ n\ \gamma \longrightarrow eval\ cid\ t\ t'\ n\ \gamma'$

proof cases
assume $(\exists i. \|cid\|_t i)$
show *?thesis*

proof cases
assume $\exists i \geq n. \|cid\|_t i$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \longrightarrow^b \gamma') \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *imp-def* **by** *simp*
ultimately have $\gamma\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (the\text{-}enat\ \langle cid\ \#_{enat}\ n\ inf\text{-}llist\ t \rangle)$
 $\longrightarrow \gamma'\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (the\text{-}enat\ \langle cid\ \#_{enat}\ n\ inf\text{-}llist\ t \rangle)$
using *validCE-act* [**where** $\gamma = \lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n$] **by** *blast*
with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*

next
assume $\neg (\exists i \geq n. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \longrightarrow^b \gamma') \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *imp-def* **by** *simp*
ultimately have $\gamma\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (cid \downarrow t n)$
 $\longrightarrow \gamma'\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (cid \downarrow t n)$
using *validCE-cont* [**where** $\gamma = \lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n$] $\langle \exists i. \|cid\|_t i \rangle$ **by** *blast*
with $\langle \neg (\exists i \geq n. \|cid\|_t i) \rangle \langle \exists i. \|cid\|_t i \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*

qed

next
assume $\neg (\exists i. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \longrightarrow^b \gamma') \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \longrightarrow \gamma'\ t\ n)$
using *imp-def* **by** *simp*

ultimately have γ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) n
 $\longrightarrow \gamma'$ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) n
using *validCE-not-act*[**where** $\gamma = \lambda t n. \gamma t n \longrightarrow \gamma' t n$] **by** *blast*
with $\langle \neg(\exists i. \|cid\|_t i) \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*
qed

2.4.4 Disjunction

definition *disj* :: ('cmp bta) \Rightarrow ('cmp bta) \Rightarrow ('cmp bta) (**infixl** \vee^b 15)
where $\gamma \vee^b \gamma' \equiv \lambda t n. \gamma t n \vee \gamma' t n$

lemma *disjI*[*intro!*]:
assumes *eval cid t t' n* $\gamma \vee \text{eval cid t t' n } \gamma'$
shows *eval cid t t' n* $(\gamma \vee^b \gamma')$
using *assms disj-def eval-def* **by** *auto*

lemma *disjE*[*elim!*]:
assumes *eval cid t t' n* $(\gamma \vee^b \gamma')$
shows *eval cid t t' n* $\gamma \vee \text{eval cid t t' n } \gamma'$

proof cases

assume $(\exists i. \|cid\|_t i)$

show *?thesis*

proof cases

assume $\exists i \geq n. \|cid\|_t i$

moreover from $\langle \text{eval cid t t' n } (\gamma \vee^b \gamma') \rangle$ **have** *eval cid t t' n* $(\lambda t n. \gamma t n \vee \gamma' t n)$

using *disj-def* **by** *simp*

ultimately have γ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) (*the-enat* $\langle cid \#_{\text{enat } n} \text{inf-llist } t \rangle$)
 $\vee \gamma'$ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) (*the-enat* $\langle cid \#_{\text{enat } n} \text{inf-llist } t \rangle$)

using *validCE-act*[**where** $\gamma = \lambda t n. \gamma t n \vee \gamma' t n$] **by** *blast*

with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **show** *?thesis*

using *validCI-act*[*of n cid t* $\gamma t'$] *validCI-act*[*of n cid t* $\gamma' t'$] **by** *blast*

next

assume $\neg(\exists i \geq n. \|cid\|_t i)$

moreover from $\langle \text{eval cid t t' n } (\gamma \vee^b \gamma') \rangle$ **have** *eval cid t t' n* $(\lambda t n. \gamma t n \vee \gamma' t n)$

using *disj-def* **by** *simp*

ultimately have γ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) (*cid* \downarrow *tn*)
 $\vee \gamma'$ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) (*cid* \downarrow *tn*)

using *validCE-cont*[**where** $\gamma = \lambda t n. \gamma t n \vee \gamma' t n$] $\langle \exists i. \|cid\|_t i \rangle$ **by** *blast*

with $\langle \neg(\exists i \geq n. \|cid\|_t i) \rangle$ $\langle \exists i. \|cid\|_t i \rangle$ **show** *?thesis*

using *validCI-cont*[*of cid t n* $\gamma t'$] *validCI-cont*[*of cid t n* $\gamma' t'$] **by** *blast*

qed

next

assume $\neg(\exists i. \|cid\|_t i)$

moreover from $\langle \text{eval cid t t' n } (\gamma \vee^b \gamma') \rangle$ **have** *eval cid t t' n* $(\lambda t n. \gamma t n \vee \gamma' t n)$

using *disj-def* **by** *simp*

ultimately have γ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) n
 $\vee \gamma'$ ($\text{lnth } (\pi_{cid} \text{inf-llist } t @_l \text{inf-llist } t')$) n

using *validCE-not-act*[**where** $\gamma = \lambda t n. \gamma t n \vee \gamma' t n$] **by** *blast*

with $\langle \neg(\exists i. \|cid\|_t i) \rangle$ **show** *?thesis*

using *validCI-not-act*[*of cid t* $\gamma t' n$] *validCI-not-act*[*of cid t* $\gamma' t' n$] **by** *blast*

qed

2.4.5 Conjunction

definition *conj* :: ('cmp bta) \Rightarrow ('cmp bta) \Rightarrow ('cmp bta) (**infixl** \wedge^b 20)
where $\gamma \wedge^b \gamma' \equiv \lambda t n. \gamma t n \wedge \gamma' t n$

lemma *conjI*[*intro!*]:

assumes $eval\ cid\ t\ t'\ n\ \gamma \wedge eval\ cid\ t\ t'\ n\ \gamma'$
 shows $eval\ cid\ t\ t'\ n\ (\gamma \wedge^b \gamma')$

proof *cases*

assume $\exists i. \|cid\|_t\ i$

show *?thesis*

proof *cases*

assume $\exists i \geq n. \|cid\|_t\ i$

with $\langle eval\ cid\ t\ t'\ n\ \gamma \wedge eval\ cid\ t\ t'\ n\ \gamma' \rangle$

have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$
 $\wedge\ \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$

using *eval-def* by *blast*

with $\langle \exists i \geq n. \|cid\|_t\ i \rangle$ have $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$

using *validCI-act*[**where** $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] by *blast*

thus *?thesis* using *conj-def* by *simp*

next

assume $\neg (\exists i \geq n. \|cid\|_t\ i)$

with $\langle \exists i. \|cid\|_t\ i \rangle$ $\langle eval\ cid\ t\ t'\ n\ \gamma \wedge eval\ cid\ t\ t'\ n\ \gamma' \rangle$

have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow t\ n)$
 $\wedge\ \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow t\ n)$ using *eval-def* by *blast*

with $\langle \exists i. \|cid\|_t\ i \rangle$ $\langle \neg (\exists i \geq n. \|cid\|_t\ i) \rangle$ have $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$

using *validCI-cont*[**where** $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] by *blast*

thus *?thesis* using *conj-def* by *simp*

qed

next

assume $\neg (\exists i. \|cid\|_t\ i)$

with $\langle eval\ cid\ t\ t'\ n\ \gamma \wedge eval\ cid\ t\ t'\ n\ \gamma' \rangle$ have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n$
 $\wedge\ \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n$ using *eval-def* by *blast*

with $\langle \neg (\exists i. \|cid\|_t\ i) \rangle$ have $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$

using *validCI-not-act*[**where** $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] by *blast*

thus *?thesis* using *conj-def* by *simp*

qed

lemma *conjE*[*elim!*]:

assumes $eval\ cid\ t\ t'\ n\ (\gamma \wedge^b \gamma')$

shows $eval\ cid\ t\ t'\ n\ \gamma \wedge eval\ cid\ t\ t'\ n\ \gamma'$

proof *cases*

assume $(\exists i. \|cid\|_t\ i)$

show *?thesis*

proof *cases*

assume $\exists i \geq n. \|cid\|_t\ i$

moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \wedge^b \gamma') \rangle$ have $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$

using *conj-def* by *simp*

ultimately have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$
 $\wedge\ \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$

using *validCE-act*[**where** $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] by *blast*

with $\langle \exists i \geq n. \|cid\|_t\ i \rangle$ show *?thesis* using *eval-def* by *blast*

next

assume $\neg (\exists i \geq n. \|cid\|_t\ i)$

moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \wedge^b \gamma') \rangle$ have $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$

using *conj-def* by *simp*

ultimately have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow t\ n)$

$\wedge\ \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow t\ n)$

using *validCE-cont*[**where** $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] $\langle \exists i. \|cid\|_t\ i \rangle$ by *blast*

with $\langle \neg (\exists i \geq n. \|cid\|_t i) \rangle \langle \exists i. \|cid\|_t i \rangle$ **show** *?thesis using eval-def by blast*
qed
next
 assume $\neg (\exists i. \|cid\|_t i)$
 moreover from $\langle eval\ cid\ t\ t'\ n\ (\gamma \wedge^b \gamma') \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n)$
 using *conj-def by simp*
 ultimately have $\gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n \wedge \gamma'\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n$
 using *validCE-not-act[where $\gamma = \lambda t\ n. \gamma\ t\ n \wedge \gamma'\ t\ n$] by blast*
 with $\langle \neg (\exists i. \|cid\|_t i) \rangle$ **show** *?thesis using eval-def by blast*
qed

2.4.6 Negation

definition $neg :: ('cmp\ bta) \Rightarrow ('cmp\ bta)\ (\neg^b - [19]\ 19)$

where $\neg^b \gamma \equiv \lambda t\ n. \neg \gamma\ t\ n$

lemma *negI[intro!]*:

assumes $\neg\ eval\ cid\ t\ t'\ n\ \gamma$

shows $eval\ cid\ t\ t'\ n\ (\neg^b \gamma)$

proof cases

assume $\exists i. \|cid\|_t i$

show *?thesis*

proof cases

assume $\exists i \geq n. \|cid\|_t i$

with $\langle \neg\ eval\ cid\ t\ t'\ n\ \gamma \rangle$

have $\neg \gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$

using *eval-def by blast*

with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \neg \gamma\ t\ n)$

using *validCI-act[where $\gamma = \lambda t\ n. \neg \gamma\ t\ n$] by blast*

thus *?thesis using neg-def by simp*

next

assume $\neg (\exists i \geq n. \|cid\|_t i)$

with $\langle \exists i. \|cid\|_t i \rangle \langle \neg\ eval\ cid\ t\ t'\ n\ \gamma \rangle$

have $\neg \gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow t n)$ **using** *eval-def by blast*

with $\langle \exists i. \|cid\|_t i \rangle \langle \neg (\exists i \geq n. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \neg \gamma\ t\ n)$

using *validCI-cont[where $\gamma = \lambda t\ n. \neg \gamma\ t\ n$] by blast*

thus *?thesis using neg-def by simp*

qed

next

assume $\neg (\exists i. \|cid\|_t i)$

with $\langle \neg\ eval\ cid\ t\ t'\ n\ \gamma \rangle$ **have** $\neg \gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n$ **using** *eval-def by blast*

with $\langle \neg (\exists i. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \neg \gamma\ t\ n)$

using *validCI-not-act[where $\gamma = \lambda t\ n. \neg \gamma\ t\ n$] by blast*

thus *?thesis using neg-def by simp*

qed

lemma *negE[elim!]*:

assumes $eval\ cid\ t\ t'\ n\ (\neg^b \gamma)$

shows $\neg\ eval\ cid\ t\ t'\ n\ \gamma$

proof cases

assume $(\exists i. \|cid\|_t i)$

show *?thesis*

proof cases

assume $\exists i \geq n. \|cid\|_t i$

moreover from $\langle eval\ cid\ t\ t'\ n\ (\neg^b \gamma) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. \neg \gamma\ t\ n)$ **using** *neg-def by simp*

ultimately have $\neg \gamma\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$

using *validCE-act*[**where** $\gamma = \lambda t n. \neg \gamma t n$] **by** *blast*
with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **show** *?thesis using eval-def by blast*
next
assume $\neg (\exists i \geq n. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\neg^b \gamma) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t n. \neg \gamma t n)$ **using** *neg-def by simp*
ultimately have $\neg \gamma (lnth (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t')) (cid \downarrow t n)$
using *validCE-cont*[**where** $\gamma = \lambda t n. \neg \gamma t n$] $\langle \exists i. \|cid\|_t i \rangle$ **by** *blast*
with $\langle \neg (\exists i \geq n. \|cid\|_t i) \rangle$ $\langle \exists i. \|cid\|_t i \rangle$ **show** *?thesis using eval-def by blast*
qed
next
assume $\neg (\exists i. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\neg^b \gamma) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t n. \neg \gamma t n)$ **using** *neg-def by simp*
ultimately have $\neg \gamma (lnth (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t')) n$
using *validCE-not-act*[**where** $\gamma = \lambda t n. \neg \gamma t n$] **by** *blast*
with $\langle \neg (\exists i. \|cid\|_t i) \rangle$ **show** *?thesis using eval-def by blast*
qed

2.4.7 Quantifiers

definition *all* :: $(a \Rightarrow (cmp\ bta))$
 $\Rightarrow (cmp\ bta)$ (**binder** $\forall b\ 10$)
where $all\ P \equiv \lambda t n. (\forall y. (P\ y\ t\ n))$

lemma *allI*[*intro!*]:

assumes $\forall p. eval\ cid\ t\ t'\ n\ (\gamma\ p)$
shows $eval\ cid\ t\ t'\ n\ (all\ (\lambda p. \gamma\ p))$

proof *cases*

assume $\exists i. \|cid\|_t i$

show *?thesis*

proof *cases*

assume $\exists i \geq n. \|cid\|_t i$
with $\langle \forall p. eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$
have $\forall p. (\gamma\ p) (lnth (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t')) (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$
using *eval-def by blast*
with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t n. (\forall y. (\gamma\ y\ t\ n)))$
using *validCI-act*[**where** $\gamma = \lambda t n. (\forall y. (\gamma\ y\ t\ n))$] **by** *blast*
thus *?thesis using all-def[of γ] by auto*

next

assume $\neg (\exists i \geq n. \|cid\|_t i)$
with $\langle \exists i. \|cid\|_t i \rangle$ $\langle \forall p. eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$
have $\forall p. (\gamma\ p) (lnth (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t')) (cid \downarrow t n)$
using *eval-def by blast*
with $\langle \exists i. \|cid\|_t i \rangle$ $\langle \neg (\exists i \geq n. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t n. (\forall y. (\gamma\ y\ t\ n)))$
using *validCI-cont*[**where** $\gamma = \lambda t n. (\forall y. (\gamma\ y\ t\ n))$] **by** *blast*
thus *?thesis using all-def[of γ] by auto*

qed

next

assume $\neg (\exists i. \|cid\|_t i)$
with $\langle \forall p. eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$ **have** $\forall p. (\gamma\ p) (lnth (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t')) n$
using *eval-def by blast*
with $\langle \neg (\exists i. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t n. (\forall y. (\gamma\ y\ t\ n)))$
using *validCI-not-act*[**where** $\gamma = \lambda t n. (\forall y. (\gamma\ y\ t\ n))$] **by** *blast*
thus *?thesis using all-def[of γ] by auto*

qed

lemma *allE*[*elim!*]:

assumes $eval\ cid\ t\ t'\ n\ (all\ (\lambda p.\ \gamma\ p))$
shows $\forall p.\ eval\ cid\ t\ t'\ n\ (\gamma\ p)$
proof cases
assume $(\exists i.\ \|cid\|_t\ i)$
show *?thesis*
proof cases
assume $\exists i \geq n.\ \|cid\|_t\ i$
moreover from $\langle eval\ cid\ t\ t'\ n\ (all\ (\lambda p.\ \gamma\ p)) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n)))$
using *all-def[of γ]* **by** *auto*
ultimately have $\forall p.\ (\gamma\ p)\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$
using *validCE-act[where $\gamma = \lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n))$]* **by** *blast*
with $\langle \exists i \geq n.\ \|cid\|_t\ i \rangle$ **show** *?thesis using eval-def by blast*
next
assume $\neg (\exists i \geq n.\ \|cid\|_t\ i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (all\ (\lambda p.\ \gamma\ p)) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n)))$
using *all-def[of γ]* **by** *auto*
ultimately have $\forall p.\ (\gamma\ p)\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow_t n)$
using *validCE-cont[where $\gamma = \lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n))$]* $\langle \exists i.\ \|cid\|_t\ i \rangle$ **by** *blast*
with $\langle \neg (\exists i \geq n.\ \|cid\|_t\ i) \rangle$ $\langle \exists i.\ \|cid\|_t\ i \rangle$ **show** *?thesis using eval-def by blast*
qed
next
assume $\neg (\exists i.\ \|cid\|_t\ i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (all\ (\lambda p.\ \gamma\ p)) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n)))$
using *all-def[of γ]* **by** *auto*
ultimately have $\forall p.\ (\gamma\ p)\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ n$
using *validCE-not-act[where $\gamma = \lambda t\ n.\ (\forall y.\ (\gamma\ y\ t\ n))$]* **by** *blast*
with $\langle \neg (\exists i.\ \|cid\|_t\ i) \rangle$ **show** *?thesis using eval-def by blast*
qed

definition $ex :: ('a \Rightarrow ('cmp\ bta))$
 $\Rightarrow ('cmp\ bta)\ (binder\ \exists_b\ 10)$
where $ex\ P \equiv \lambda t\ n.\ (\exists y.\ (P\ y\ t\ n))$

lemma $exI[intro]$:
assumes $\exists p.\ eval\ cid\ t\ t'\ n\ (\gamma\ p)$
shows $eval\ cid\ t\ t'\ n\ (\exists_b p.\ \gamma\ p)$
proof cases
assume $\exists i.\ \|cid\|_t\ i$
show *?thesis*
proof cases
assume $\exists i \geq n.\ \|cid\|_t\ i$
with $\langle \exists p.\ eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$
have $\exists p.\ (\gamma\ p)\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (the-enat\ \langle cid\ \#_{enat}\ n\ inf-llist\ t \rangle)$
using *eval-def by blast*
with $\langle \exists i \geq n.\ \|cid\|_t\ i \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n.\ (\exists y.\ (\gamma\ y\ t\ n)))$
using *validCI-act[where $\gamma = \lambda t\ n.\ (\exists y.\ (\gamma\ y\ t\ n))$]* **by** *blast*
thus *?thesis using ex-def[of γ]* **by** *auto*
next
assume $\neg (\exists i \geq n.\ \|cid\|_t\ i)$
with $\langle \exists i.\ \|cid\|_t\ i \rangle$ $\langle \exists p.\ eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$
have $\exists p.\ (\gamma\ p)\ (lnth\ (\pi_{cid} inf-llist\ t\ @_l\ inf-llist\ t'))\ (cid \downarrow_t n)$ **using** *eval-def by blast*
with $\langle \exists i.\ \|cid\|_t\ i \rangle$ $\langle \neg (\exists i \geq n.\ \|cid\|_t\ i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n.\ (\exists y.\ (\gamma\ y\ t\ n)))$
using *validCI-cont[where $\gamma = \lambda t\ n.\ (\exists y.\ (\gamma\ y\ t\ n))$]* **by** *blast*
thus *?thesis using ex-def[of γ]* **by** *auto*
qed

next
assume $\neg(\exists i. \|cid\|_t i)$
with $\langle \exists p. eval\ cid\ t\ t'\ n\ (\gamma\ p) \rangle$ **have** $\exists p. (\gamma\ p)\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ n$
using *eval-def* **by** *blast*
with $\langle \neg(\exists i. \|cid\|_t i) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. (\exists y. (\gamma\ y\ t\ n)))$
using *validCI-not-act* **[where** $\gamma = \lambda t\ n. (\exists y. (\gamma\ y\ t\ n))$ **]** **by** *blast*
thus *?thesis* **using** *ex-def* **[of** γ **]** **by** *auto*
qed

lemma *exE[elim!]*:
assumes $eval\ cid\ t\ t'\ n\ (\exists_b p. \gamma\ p)$
shows $\exists p. eval\ cid\ t\ t'\ n\ (\gamma\ p)$
proof *cases*
assume $(\exists i. \|cid\|_t i)$
show *?thesis*
proof *cases*
assume $\exists i \geq n. \|cid\|_t i$
moreover from $\langle eval\ cid\ t\ t'\ n\ (ex\ (\lambda p. \gamma\ p)) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. (\exists y. (\gamma\ y\ t\ n)))$
using *ex-def* **[of** γ **]** **by** *auto*
ultimately have $\exists p. (\gamma\ p)\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (the\ enat\ \langle cid\ \#_{enat}\ n\ inf\text{-}llist\ t \rangle)$
using *validCE-act* **[where** $\gamma = \lambda t\ n. (\exists y. (\gamma\ y\ t\ n))$ **]** **by** *blast*
with $\langle \exists i \geq n. \|cid\|_t i \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*
next
assume $\neg(\exists i \geq n. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\exists_b p. \gamma\ p) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. (\exists y. (\gamma\ y\ t\ n)))$
using *ex-def* **[of** γ **]** **by** *auto*
ultimately have $\exists p. (\gamma\ p)\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ (cid \downarrow t\ n)$
using *validCE-cont* **[where** $\gamma = \lambda t\ n. (\exists y. (\gamma\ y\ t\ n))$ **]** $\langle \exists i. \|cid\|_t i \rangle$ **by** *blast*
with $\langle \neg(\exists i \geq n. \|cid\|_t i) \rangle$ $\langle \exists i. \|cid\|_t i \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*
qed

next
assume $\neg(\exists i. \|cid\|_t i)$
moreover from $\langle eval\ cid\ t\ t'\ n\ (\exists_b p. \gamma\ p) \rangle$ **have** $eval\ cid\ t\ t'\ n\ (\lambda t\ n. (\exists y. (\gamma\ y\ t\ n)))$
using *ex-def* **[of** γ **]** **by** *auto*
ultimately have $\exists p. (\gamma\ p)\ (lnth\ (\pi_{cid} inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ n$
using *validCE-not-act* **[where** $\gamma = \lambda t\ n. (\exists y. (\gamma\ y\ t\ n))$ **]** **by** *blast*
with $\langle \neg(\exists i. \|cid\|_t i) \rangle$ **show** *?thesis* **using** *eval-def* **by** *blast*
qed

2.4.8 Behavior Assertions

First we provide rules for basic behavior assertions.

definition $ba :: ('cmp \Rightarrow bool) \Rightarrow ('cmp\ bta)\ ([-]_b)$
where $ba\ \varphi \equiv \lambda t\ n. \varphi\ (t\ n)$

lemma *baIA[intro]*:
fixes $c :: 'id$
and $t :: nat \Rightarrow cnf$
and $t' :: nat \Rightarrow 'cmp$
and $n :: nat$
assumes $\exists i \geq n. \|c\|_t i$
and $\varphi\ (\sigma_c(t\ \langle c \rightarrow t \rangle n))$
shows $eval\ c\ t\ t'\ n\ (ba\ \varphi)$
proof $-$
from *assms* **have** $\varphi\ (\sigma_c(t\ \langle c \rightarrow t \rangle n))$ **by** *simp*

moreover have $\sigma_c(t \langle c \rightarrow t \rangle_n) = \text{lnth } (\pi_c(\text{inf-llist } t)) \text{ (the-enat } (\langle c \#_{\langle c \rightarrow t \rangle_n} \text{ inf-llist } t))$

proof –

have $\text{enat } (\text{Suc } \langle c \rightarrow t \rangle_n) < \text{llength } (\text{inf-llist } t)$ **using** enat-ord-code **by** simp

moreover from assms have $\|c\|_t (\langle c \rightarrow t \rangle_n)$ **using** nxtActI **by** simp

hence $\|c\|_{\text{lnth } (\text{inf-llist } t) \langle c \rightarrow t \rangle_n}$ **by** simp

ultimately show $?thesis$ **using** proj-active-nth **by** simp

qed

ultimately have $\varphi (\text{lnth } (\pi_c(\text{inf-llist } t)) \text{ (the-enat } (\langle c \#_{\langle c \rightarrow t \rangle_n} \text{ inf-llist } t)))$ **by** simp

moreover have $\langle c \#_n \text{ inf-llist } t \rangle = \langle c \#_{\langle c \rightarrow t \rangle_n} \text{ inf-llist } t \rangle$

proof –

from assms have $\nexists k. n \leq k \wedge k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k$ **using** nxtActI **by** simp

hence $\neg (\exists k \geq n. k < \langle c \rightarrow t \rangle_n \wedge \|c\|_{\text{lnth } (\text{inf-llist } t) k})$ **by** simp

moreover have $\text{enat } \langle c \rightarrow t \rangle_n - 1 < \text{llength } (\text{inf-llist } t)$ **by** $(\text{simp add: one-enat-def})$

moreover from assms have $\langle c \rightarrow t \rangle_n \geq n$ **using** nxtActI **by** simp

ultimately show $?thesis$ **using** $\text{nAct-not-active-same}$ [of $n \langle c \rightarrow t \rangle_n \text{ inf-llist } t \ c$] **by** simp

qed

ultimately have $\varphi (\text{lnth } (\pi_c(\text{inf-llist } t)) \text{ (the-enat } (\langle c \#_n \text{ inf-llist } t)))$ **by** simp

moreover have $\text{enat } (\text{the-enat } (\langle c \#_{\text{enat } n} \text{ inf-llist } t)) < \text{llength } (\pi_c(\text{inf-llist } t))$

proof –

have $\text{ltake } \infty (\text{inf-llist } t) = (\text{inf-llist } t)$ **using** ltake-all [of $\text{inf-llist } t$] **by** simp

hence $\text{llength } (\pi_c(\text{inf-llist } t)) = \langle c \#_{\infty} \text{ inf-llist } t \rangle$ **using** nAct-def **by** simp

moreover have $\langle c \#_{\text{enat } n} \text{ inf-llist } t \rangle < \langle c \#_{\infty} \text{ inf-llist } t \rangle$

proof –

have $\text{enat } \langle c \rightarrow t \rangle_n < \text{llength } (\text{inf-llist } t)$ **by** simp

moreover from assms have $\langle c \rightarrow t \rangle_n \geq n$ **and** $\|c\|_t (\langle c \rightarrow t \rangle_n)$ **using** nxtActI **by** auto

ultimately show $?thesis$ **using** nAct-less [of $\langle c \rightarrow t \rangle_n \text{ inf-llist } t \ n \ \infty$] **by** simp

qed

ultimately show $?thesis$ **by** simp

qed

hence $\text{lnth } (\pi_c(\text{inf-llist } t)) \text{ (the-enat } (\langle c \#_n \text{ inf-llist } t)) =$

$\text{lnth } ((\pi_c(\text{inf-llist } t)) \ @_l (\text{inf-llist } t')) \text{ (the-enat } (\langle c \#_n \text{ inf-llist } t))$

using lnth-lappendI [of $\text{the-enat } (\langle c \#_{\text{enat } n} \text{ inf-llist } t) \ \pi_c(\text{inf-llist } t) \ \text{inf-llist } t'$] **by** simp

ultimately have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) \ @_l (\text{inf-llist } t')) \text{ (the-enat } (\langle c \#_n \text{ inf-llist } t)))$ **by** simp

hence $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) \ @_l (\text{inf-llist } t')) \text{ (the-enat } (\langle c \#_n \text{ inf-llist } t)))$ **by** simp

moreover from assms have $\langle c \rightarrow t \rangle_n \geq n$ **and** $\|c\|_t (\langle c \rightarrow t \rangle_n)$ **using** nxtActI **by** auto

ultimately have $(\exists i \geq \text{snd } (t, n). \|c\|_{\text{fst } (t, n) i} \wedge$

$\varphi (\text{lnth } ((\pi_c(\text{inf-llist } (\text{fst } (t, n)))) \ @_l (\text{inf-llist } t'))$

$\text{ (the-enat } (\langle c \#_{\text{the-enat } (\text{snd } (t, n))} \text{ inf-llist } (\text{fst } (t, n))))$) **by** auto

thus $?thesis$ **using** ba-def **by** simp

qed

lemma baIN1 [intro]:

fixes $c::'id$

and $t::\text{nat} \Rightarrow \text{cnf}$

and $t'::\text{nat} \Rightarrow 'cmp$

and $n::\text{nat}$

assumes $\text{act: } \exists i. \|c\|_t i$

and $\text{nAct: } \nexists i. i \geq n \wedge \|c\|_t i$

and $\text{al: } \varphi (t' (n - \langle c \wedge t \rangle - 1))$

shows $\text{eval } c \ t \ t' \ n$ (ba φ)

proof –

have $t' (n - \langle c \wedge t \rangle - 1) = \text{lnth } (\text{inf-llist } t') (n - \langle c \wedge t \rangle - 1)$ **by** simp

moreover have $\dots = \text{lnth } ((\pi_c(\text{inf-llist } t)) \ @_l (\text{inf-llist } t')) (c \downarrow_t (n))$

using $\text{act nAct cnf2bhv-lnth-lappend}$ **by** simp

ultimately have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (c \downarrow_t(n)))$ using *al* by *simp*
with *act nAct* show *?thesis* using *ba-def* by *simp*
qed

lemma *baIN2[intro]*:

fixes *c::'id*
and *t::nat* \Rightarrow *cnf*
and *t'::nat* \Rightarrow *'cmp*
and *n::nat*
assumes *nAct*: $\# i. \|c\|_t i$
and *al*: $\varphi (t' n)$
shows *eval c t t' n (ba φ)*

proof –

have *t' n = lnth (inf-llist t') n* by *simp*
moreover have $\dots = \text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) n$

proof –

from *nAct* have $\pi_c(\text{inf-llist } t) = []_l$ by *simp*
hence $(\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t') = \text{inf-llist } t'$ by (*simp add: <math>\langle \pi_c \text{inf-llist } t = []_l \rangle)
thus *?thesis* by *simp**

qed

ultimately have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) n)$ using *al* by *simp*
hence $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) n)$ by *simp*
with *nAct* show *?thesis* using *ba-def* by *simp*

qed

lemma *baIANow[intro]*:

fixes *t n c φ*
assumes $\varphi (\sigma_c(t n))$
and $\|c\|_t n$
shows *eval c t t' n (ba φ)*

proof –

from *assms* have $\varphi(\sigma_c(t \langle c \rightarrow t \rangle_n))$ using *nxtAct-active* by *simp*
with *assms* show *?thesis* using *baIA* by *blast*

qed

lemma *baEA[elim]*:

fixes *c::'id*
and *t::nat* \Rightarrow *cnf*
and *t'::nat* \Rightarrow *'cmp*
and *n::nat*
and *i::nat*
assumes $\exists i \geq n. \|c\|_t i$
and *eval c t t' n (ba φ)*
shows $\varphi (\sigma_c(t \langle c \rightarrow t \rangle_n))$

proof –

from $\langle \text{eval } c \text{ t t' n (ba } \varphi) \rangle$ have *eval c t t' n* ($\lambda t n. \varphi (t n)$) using *ba-def* by *simp*
moreover from *assms* have $\langle c \rightarrow t \rangle_{n \geq n}$ and $\|c\|_t (\langle c \rightarrow t \rangle_n)$ using *nxtActI[of n c t]* by *auto*
ultimately have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)))$
using *validCE-act* by *blast*
hence $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)))$ by *simp*
moreover have *enat (the-enat (<math>\langle c \#_{\text{enat } n} \text{inf-llist } t \rangle)) < \text{llength } (\pi_c(\text{inf-llist } t))*

proof –

have *ltake ∞ (inf-llist t) = (inf-llist t)* using *ltake-all[of inf-llist t]* by *simp*
hence $\text{llength } (\pi_c(\text{inf-llist } t)) = \langle c \#_\infty \text{inf-llist } t \rangle$ using *nAct-def* by *simp*
moreover have $\langle c \#_{\text{enat } n} \text{inf-llist } t \rangle < \langle c \#_\infty \text{inf-llist } t \rangle$

proof –
 have $\text{enat } \langle c \rightarrow t \rangle_n < \text{llength } (\text{inf-llist } t)$ **by simp**
 with $\langle \langle c \rightarrow t \rangle_{n \geq n} \rangle \langle \|c\|_t \langle c \rightarrow t \rangle_n \rangle$ **show ?thesis using nAct-less by simp**
qed
 ultimately show ?thesis **by simp**
qed
 hence $\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)) =$
 $\text{lnth } (\pi_c(\text{inf-llist } t)) (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle))$
 using lnth-lappend1 [of $\text{the-enat } (\langle c \#_{\text{enat } n} \text{inf-llist } t) \pi_c(\text{inf-llist } t) \text{inf-llist } t'$] **by simp**
 ultimately have $\varphi (\text{lnth } (\pi_c(\text{inf-llist } t)) (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)))$ **by simp**
 moreover have $\langle c \#_n \text{inf-llist } t \rangle = \langle c \#_{\langle c \rightarrow t \rangle_n} \text{inf-llist } t \rangle$
proof –
 from assms have $\nexists k. n \leq k \wedge k < \langle c \rightarrow t \rangle_n \wedge \|c\|_t k$ **using nextActI** [of $n \ c \ t$] **by auto**
 hence $\neg (\exists k \geq n. k < \langle c \rightarrow t \rangle_n \wedge \|c\|_{\text{lnth } (\text{inf-llist } t) \ k})$ **by simp**
 moreover have $\text{enat } \langle c \rightarrow t \rangle_n - 1 < \text{llength } (\text{inf-llist } t)$ **by (simp add: one-enat-def)**
 ultimately show ?thesis **using** $\langle \langle c \rightarrow t \rangle_{n \geq n} \rangle$ $n\text{Act-not-active-same}$ **by simp**
qed
 moreover have $\sigma_c(t \langle c \rightarrow t \rangle_n) = \text{lnth } (\pi_c(\text{inf-llist } t)) (\text{the-enat } (\langle c \#_{\langle c \rightarrow t \rangle_n} \text{inf-llist } t))$
proof –
 have $\text{enat } (\text{Suc } i) < \text{llength } (\text{inf-llist } t)$ **using enat-ord-code by simp**
 moreover from $\langle \|c\|_t \langle c \rightarrow t \rangle_n \rangle$ have $\|c\|_{\text{lnth } (\text{inf-llist } t) \ \langle c \rightarrow t \rangle_n}$ **by simp**
 ultimately show ?thesis **using proj-active-nth by simp**
qed
 ultimately show ?thesis **by simp**
qed

lemma $\text{baEN1}[\text{elim}]$:
 fixes $c::'id$
 and $t::\text{nat} \Rightarrow \text{cnf}$
 and $t'::\text{nat} \Rightarrow 'cmp$
 and $n::\text{nat}$
 assumes $\text{act}: \exists i. \|c\|_t i$
 and $n\text{Act}: \nexists i. i \geq n \wedge \|c\|_t i$
 and $\text{al}: \text{eval } c \ t \ t' \ n \ (\text{ba } \varphi)$
 shows $\varphi (t' (n - \langle c \wedge t \rangle - 1))$
proof –
 from al have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (c \downarrow_t (n)))$
 using $\text{act } n\text{Act } \text{validCE-cont } \text{ba-def}$ **by metis**
 hence $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (c \downarrow_t (n)))$ **by simp**
 moreover have $\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) (c \downarrow_t (n)) = \text{lnth } (\text{inf-llist } t') (n - \langle c \wedge t \rangle - 1)$
 using $\text{act } n\text{Act } \text{cnf2bhv-lnth-lappend}$ **by simp**
 moreover have $\dots = t' (n - \langle c \wedge t \rangle - 1)$ **by simp**
 ultimately show ?thesis **by simp**
qed

lemma $\text{baEN2}[\text{elim}]$:
 fixes $c::'id$
 and $t::\text{nat} \Rightarrow \text{cnf}$
 and $t'::\text{nat} \Rightarrow 'cmp$
 and $n::\text{nat}$
 assumes $n\text{Act}: \nexists i. \|c\|_t i$
 and $\text{al}: \text{eval } c \ t \ t' \ n \ (\text{ba } \varphi)$
 shows $\varphi (t' n)$
proof –
 from al have $\varphi (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')) n)$

using $nAct$ *validCE-not-act ba-def* by *metis*
 hence φ ($lnth$ ($(\pi_c(inf-llist\ t)) @_l (inf-llist\ t')$) n) by *simp*
 moreover have $lnth$ ($(\pi_c(inf-llist\ t)) @_l (inf-llist\ t')$) $n = lnth$ ($inf-llist\ t'$) n
 proof –
 from $nAct$ have $\pi_c(inf-llist\ t) = []_l$ by *simp*
 hence $(\pi_c(inf-llist\ t)) @_l (inf-llist\ t') = inf-llist\ t'$ by (*simp add: <math>\langle \pi_c inf-llist\ t = []_l \rangle)
 thus *?thesis* by *simp*
 qed
 moreover have $\dots = t' n$ by *simp*
 ultimately show *?thesis* by *simp*
 qed*

lemma *baEANow[elim]*:

fixes $t\ n\ c\ \varphi$
 assumes *eval* $c\ t\ t'\ n$ (*ba* φ)
 and $\|c\|_t\ n$
 shows φ ($\sigma_c(t\ n)$)

proof –
 from *assms* have $\varphi(\sigma_c(t\ \langle c \rightarrow t \rangle_n))$ using *baEA* by *blast*
 with *assms* show *?thesis* using *nxtAct-active* by *simp*
 qed

2.4.9 Next Operator

definition *nxt* :: $(\text{'cmp}\ bta) \Rightarrow (\text{'cmp}\ bta) (\circ_b(-)\ 24)$
 where $\circ_b(\gamma) \equiv \lambda\ t\ n.\ \gamma\ t\ (Suc\ n)$

lemma *nxtIA[intro]*:

fixes $c::'id$
 and $t::nat \Rightarrow cnf$
 and $t':nat \Rightarrow \text{'cmp}$
 and $n::nat$
 assumes $\exists i \geq n.\ \|c\|_t\ i$
 and $[\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i] \Longrightarrow \exists n' \geq n.\ (\exists! i.\ n \leq i \wedge i < n' \wedge \|c\|_t\ i) \wedge eval\ c\ t\ t'\ n'\ \gamma$
 and $[\neg(\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i)] \Longrightarrow eval\ c\ t\ t'\ (Suc\ \langle c \rightarrow t \rangle_n)\ \gamma$
 shows *eval* $c\ t\ t'\ n$ ($\circ_b(\gamma)$)

proof (*cases*)

assume $\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i$
 with *assms*(2) obtain n' where $n' \geq n$ and $\exists! i.\ n \leq i \wedge i < n' \wedge \|c\|_t\ i$ and *eval* $c\ t\ t'\ n'\ \gamma$ by *blast*
 moreover from *assms*(1) have $\|c\|_t\ \langle c \rightarrow t \rangle_n$ and $\langle c \rightarrow t \rangle_n \geq n$ using *nxtActI* by *auto*
 ultimately have $\exists i' \geq n'. \|c\|_t\ i'$ by (*metis* $\langle \exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i \rangle$ *dual-order.strict-trans2 leI nat-less-le*)
 with $\langle eval\ c\ t\ t'\ n'\ \gamma \rangle$
 have γ ($lnth$ ($(\pi_c(inf-llist\ t)) @_l (inf-llist\ t')$)) (*the-enat* ($\langle c \#_{enat\ n'} inf-llist\ t \rangle$))
 using *validCE-act* by *blast*
 moreover have *the-enat*($\langle c \#_{n'} inf-llist\ t \rangle$) = *Suc* (*the-enat* ($\langle c \#_n inf-llist\ t \rangle$))
 proof –
 from $\langle \exists! i.\ n \leq i \wedge i < n' \wedge \|c\|_t\ i \rangle$ obtain i where $n \leq i$ and $i < n'$ and $\|c\|_t\ i$
 and $\forall i'. n \leq i' \wedge i' < n' \wedge \|c\|_t\ i' \longrightarrow i' = i$ by *blast*
 moreover have $n' - 1 < llength$ ($inf-llist\ t$) by *simp*
 ultimately have *the-enat*($\langle c \#_{n'} inf-llist\ t \rangle$) = *the-enat*(*eSuc* ($\langle c \#_n inf-llist\ t \rangle$))
 using *nAct-active-suc*[of $inf-llist\ t\ n'\ n\ i\ c$] by (*simp add: <math>\langle n \leq i \rangle)
 moreover have $\langle c \#_i inf-llist\ t \rangle \neq \infty$ by *simp*
 ultimately show *?thesis* using *the-enat-eSuc* by *simp*
 qed
 ultimately have γ ($lnth$ ($(\pi_c(inf-llist\ t)) @_l (inf-llist\ t')$)) (*Suc* (*the-enat* ($\langle c \#_n inf-llist\ t \rangle$))))
 by *simp**

with *assms* **have** $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$
using $validCI\text{-}act[of\ n\ c\ t\ \lambda t\ n.\ \gamma\ t\ (Suc\ n)\ t']$ **by** *blast*
thus *?thesis* **using** *nxt-def* **by** *simp*
next
assume $\neg(\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i)$
with *assms*(β) **have** $eval\ c\ t\ t'\ (Suc\ \langle c \rightarrow t \rangle_n)\ \gamma$ **by** *simp*
moreover from $\langle \neg(\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i) \rangle$ **have** $\neg(\exists i \geq Suc\ \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i)$ **by** *simp*
ultimately have $\gamma\ (lnth\ (\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (c\downarrow_t(Suc\ \langle c \rightarrow t \rangle_n))$
using *assms*(1) $validCE\text{-}cont[of\ c\ t\ Suc\ \langle c \rightarrow t \rangle_n\ t'\ \gamma]$ **by** *blast*
moreover from *assms*(1) $\langle \neg(\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i) \rangle$
have $Suc\ (the\text{-}enat\ \langle c\ \#_{enat}\ n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle) = c\downarrow_t(Suc\ \langle c \rightarrow t \rangle_n)$
using *nAct-cnf2proj-Suc-dist* **by** *simp*
ultimately have $\gamma\ (lnth\ ((\pi_c(inf\text{-}l\!l\!i\!s\!t\ t))\ @_l\ (inf\text{-}l\!l\!i\!s\!t\ t')))\ (Suc\ (the\text{-}enat\ (\langle c\ \#_n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle)))$
by *simp*
moreover from *assms*(1) **have** $\|c\|_t\ \langle c \rightarrow t \rangle_n$ **and** $\langle c \rightarrow t \rangle_n \geq n$ **using** *nxtActI* **by** *auto*
ultimately have $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$ **using** $validCI\text{-}act[of\ n\ c\ t\ \lambda t\ n.\ \gamma\ t\ (Suc\ n)\ t']$
by *blast*
with $\langle \|c\|_t\ \langle c \rightarrow t \rangle_n \rangle\ \langle \neg(\exists i' \geq Suc\ \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i') \rangle$ **show** *?thesis* **using** *nxt-def* **by** *simp*
qed

lemma *nxtIN*[*intro*]:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

assumes $\neg(\exists i \geq n.\ \|c\|_t\ i)$

and $eval\ c\ t\ t'\ (Suc\ n)\ \gamma$

shows $eval\ c\ t\ t'\ n\ (\circ_b(\gamma))$

proof *cases*

assume $\exists i.\ \|c\|_t\ i$

moreover from $\langle \neg(\exists i \geq n.\ \|c\|_t\ i) \rangle$ **have** $\neg(\exists i \geq Suc\ n.\ \|c\|_t\ i)$ **by** *simp*

ultimately have $\gamma\ (lnth\ ((\pi_c(inf\text{-}l\!l\!i\!s\!t\ t))\ @_l\ (inf\text{-}l\!l\!i\!s\!t\ t')))\ (c\downarrow_t(Suc\ n))$

using $validCE\text{-}cont\ \langle eval\ c\ t\ t'\ (Suc\ n)\ \gamma \rangle$ **by** *blast*

with $\langle \exists i.\ \|c\|_t\ i \rangle$ **have** $\gamma\ (lnth\ ((\pi_c(inf\text{-}l\!l\!i\!s\!t\ t))\ @_l\ (inf\text{-}l\!l\!i\!s\!t\ t')))\ (Suc\ (c\downarrow_t(n)))$

using $\langle \neg(\exists i \geq n.\ \|c\|_t\ i) \rangle$ *LActive-less* **by** *auto*

with $\langle \neg(\exists i \geq n.\ \|c\|_t\ i) \rangle\ \langle \exists i.\ \|c\|_t\ i \rangle$ **have** $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$

using $validCI\text{-}cont[\mathbf{where}\ \gamma=(\lambda t\ n.\ \gamma\ t\ (Suc\ n))]$ **by** *simp*

thus *?thesis* **using** *nxt-def* **by** *simp*

next

assume $\neg(\exists i.\ \|c\|_t\ i)$

with *assms* **have** $\gamma\ (lnth\ (\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (Suc\ n)$ **using** $validCE\text{-}not\text{-}act$ **by** *blast*

with $\langle \neg(\exists i.\ \|c\|_t\ i) \rangle$ **have** $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$

using $validCI\text{-}not\text{-}act[\mathbf{where}\ \gamma=(\lambda t\ n.\ \gamma\ t\ (Suc\ n))]$ **by** *blast*

thus *?thesis* **using** *nxt-def* **by** *simp*

qed

lemma *nxtEA1*[*elim*]:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

assumes $\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i$

and $eval\ c\ t\ t'\ n\ (\circ_b(\gamma))$

and $n' \geq n$

and $\exists !i.\ i \geq n \wedge i < n' \wedge \|c\|_t\ i$

shows $eval\ c\ t\ t'\ n'\ \gamma$

proof –

from $\langle eval\ c\ t\ t'\ n\ (\circ_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$ using *nxt-def* by *simp*

moreover from *assms(4)* obtain i where $i \geq n$ and $i < n'$ and $\|c\|_t\ i$

and $\forall i'.\ n \leq i' \wedge i' < n' \wedge \|c\|_t\ i' \longrightarrow i' = i$ by *blast*

ultimately have $\gamma\ (lnth\ (\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (Suc\ (the\text{-}enat\ \langle c\ \#_{enat}\ n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle))$

using *validCE-act*[of $n\ c\ t\ t'\ \lambda t\ n.\ \gamma\ t\ (Suc\ n)$] by *blast*

moreover have $the\text{-}enat(\langle c\ \#_{n'}\ inf\text{-}l\!l\!i\!s\!t\ t \rangle) = Suc\ (the\text{-}enat\ (\langle c\ \#_n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle))$

proof –

have $n' - 1 < llength\ (inf\text{-}l\!l\!i\!s\!t\ t)$ by *simp*

with $\langle i < n' \rangle$ and $\langle \|c\|_t\ i \rangle$ and $\langle \forall i'.\ n \leq i' \wedge i' < n' \wedge \|c\|_t\ i' \longrightarrow i' = i \rangle$

have $the\text{-}enat(\langle c\ \#_{n'}\ inf\text{-}l\!l\!i\!s\!t\ t \rangle) = the\text{-}enat(eSuc\ (\langle c\ \#_n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle))$

using *nAct-active-suc*[of $inf\text{-}l\!l\!i\!s\!t\ t\ n'\ n\ i\ c$] by *(simp add: n ≤ i)*

moreover have $\langle c\ \#_i\ inf\text{-}l\!l\!i\!s\!t\ t \rangle \neq \infty$ by *simp*

ultimately show *?thesis* using *the-enat-eSuc* by *simp*

qed

ultimately have $\gamma\ (lnth\ ((\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t)\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (the\text{-}enat\ (\langle c\ \#_{n'}\ inf\text{-}l\!l\!i\!s\!t\ t \rangle))$ by *simp*

moreover have $\exists i' \geq n'.\ \|c\|_t\ i'$

proof –

from *assms(4)* have $\langle c \rightarrow t \rangle_n \geq n$ and $\|c\|_t\ \langle c \rightarrow t \rangle_n$ using *nxtActI* by *auto*

with $\langle \forall i'.\ n \leq i' \wedge i' < n' \wedge \|c\|_t\ i' \longrightarrow i' = i \rangle$ show *?thesis*

using *assms(1)* by *(metis leI le-trans less-le)*

qed

ultimately show *?thesis* using *validCI-act* by *blast*

qed

lemma *nxtEA2[elim]*:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

and i

assumes $\exists i \geq n.\ \|c\|_t\ i$ and $\neg(\exists i > \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i)$

and $eval\ c\ t\ t'\ n\ (\circ_b(\gamma))$

shows $eval\ c\ t\ t'\ (Suc\ \langle c \rightarrow t \rangle_n)\ \gamma$

proof –

from $\langle eval\ c\ t\ t'\ n\ (\circ_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n.\ \gamma\ t\ (Suc\ n))$ using *nxt-def* by *simp*

with *assms(1)* have $\gamma\ (lnth\ (\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (Suc\ (the\text{-}enat\ \langle c\ \#_{enat}\ n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle))$

using *validCE-act*[of $n\ c\ t\ t'\ \lambda t\ n.\ \gamma\ t\ (Suc\ n)$] by *blast*

moreover from *assms(1)* *assms(2)* have $Suc\ (the\text{-}enat\ \langle c\ \#_{enat}\ n\ inf\text{-}l\!l\!i\!s\!t\ t \rangle) = c \downarrow_t (Suc\ \langle c \rightarrow t \rangle_n)$

using *nAct-cnff2proj-Suc-dist* by *simp*

ultimately have $\gamma\ (lnth\ (\pi_c\ inf\text{-}l\!l\!i\!s\!t\ t\ @_l\ inf\text{-}l\!l\!i\!s\!t\ t'))\ (c \downarrow_t (Suc\ \langle c \rightarrow t \rangle_n))$ by *simp*

moreover from *assms(1)* *assms(2)* have $\neg(\exists i' \geq Suc\ \langle c \rightarrow t \rangle_n.\ \|c\|_t\ i')$

using *nxtActive-no-active* by *simp*

ultimately show *?thesis* using *validCI-cont*[where $n = Suc\ \langle c \rightarrow t \rangle_n$] *assms(1)* by *blast*

qed

lemma *nxtEN[elim]*:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

assumes $\neg(\exists i \geq n.\ \|c\|_t\ i)$

and $eval\ c\ t\ t'\ n\ (\circ_b(\gamma))$

shows $eval\ c\ t\ t'\ (Suc\ n)\ \gamma$

proof cases

assume $\exists i. \|c\|_t i$
moreover from $\langle eval\ c\ t\ t'\ n\ (\circ_b(\gamma)) \rangle$ **have** $eval\ c\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ (Suc\ n))$ **using** *next-def* **by** *simp*
ultimately have $\gamma\ (lnth\ (\pi_c\ inf\ llist\ t\ @_l\ inf\ llist\ t'))\ (Suc\ (c\downarrow_t\ n))$
using $\langle \neg(\exists i \geq n. \|c\|_t i) \rangle$ *validCE-cont* [**where** $\gamma = (\lambda t\ n. \gamma\ t\ (Suc\ n))$] **by** *simp*
hence $\gamma\ (lnth\ ((\pi_c\ (inf\ llist\ t))\ @_l\ (inf\ llist\ t')))\ (c\downarrow_t\ (Suc\ n))$
using $\langle \exists i. \|c\|_t i \rangle$ *assms(1)* *lActive-less* **by** *auto*
moreover from $\langle \neg(\exists i \geq n. \|c\|_t i) \rangle$ **have** $\neg(\exists i \geq Suc\ n. \|c\|_t i)$ **by** *simp*
ultimately show *?thesis* **using** *validCI-cont* [**where** $n = Suc\ n$] $\langle \exists i. \|c\|_t i \rangle$ **by** *blast*

next

assume $\neg(\exists i. \|c\|_t i)$
moreover from $\langle eval\ c\ t\ t'\ n\ (\circ_b(\gamma)) \rangle$ **have** $eval\ c\ t\ t'\ n\ (\lambda t\ n. \gamma\ t\ (Suc\ n))$ **using** *next-def* **by** *simp*
ultimately have $\gamma\ (lnth\ (\pi_c\ inf\ llist\ t\ @_l\ inf\ llist\ t'))\ (Suc\ n)$
using $\langle \neg(\exists i. \|c\|_t i) \rangle$ *validCE-not-act* [**where** $\gamma = (\lambda t\ n. \gamma\ t\ (Suc\ n))$] **by** *blast*
with $\langle \neg(\exists i. \|c\|_t i) \rangle$ **show** *?thesis* **using** *validCI-not-act* [*of* $c\ t\ \gamma\ t'\ Suc\ n$] **by** *blast*

qed

2.4.10 Eventually Operator

definition $evt :: ('cmp\ bta) \Rightarrow ('cmp\ bta)\ (\diamond_b(-)\ 23)$

where $\diamond_b(\gamma) \equiv \lambda t\ n. \exists n' \geq n. \gamma\ t\ n'$

lemma *evtIA* [*intro*]:

fixes $c :: 'id$

and $t :: nat \Rightarrow cnf$

and $t' :: nat \Rightarrow 'cmp$

and $n :: nat$

and $n' :: nat$

assumes $\exists i \geq n. \|c\|_t i$

and $n' \geq \langle c \Leftarrow t \rangle_n$

and $\llbracket \exists i \geq n'. \|c\|_t i \rrbracket \Longrightarrow \exists n'' \geq \langle c \Leftarrow t \rangle_{n'}. n'' \leq \langle c \rightarrow t \rangle_{n'} \wedge eval\ c\ t\ t'\ n''\ \gamma$

and $\llbracket \neg(\exists i \geq n'. \|c\|_t i) \rrbracket \Longrightarrow eval\ c\ t\ t'\ n'\ \gamma$

shows $eval\ c\ t\ t'\ n\ (\diamond_b(\gamma))$

proof cases **assume** $\exists i' \geq n'. \|c\|_t i'$

with *assms(3)* **obtain** n'' **where** $n'' \geq \langle c \Leftarrow t \rangle_{n'}$ **and** $n'' \leq \langle c \rightarrow t \rangle_{n'}$ **and** $eval\ c\ t\ t'\ n''\ \gamma$ **by** *auto*

hence $\exists i' \geq n''. \|c\|_t i'$ **using** $\langle \exists i' \geq n'. \|c\|_t i' \rangle$ *nextActI* **by** *blast*

with $\langle eval\ c\ t\ t'\ n''\ \gamma \rangle$ **have**

$\gamma\ (lnth\ ((\pi_c\ (inf\ llist\ t))\ @_l\ (inf\ llist\ t')))\ (the\ enat\ (\langle c\ \#_{n''}\ inf\ llist\ t \rangle))$

using *validCE-act* **by** *blast*

moreover have $the\ enat\ (\langle c\ \#_n\ inf\ llist\ t \rangle) \leq the\ enat\ (\langle c\ \#_{n''}\ inf\ llist\ t \rangle)$

proof –

from $\langle \langle c \Leftarrow t \rangle_{n'} \leq n'' \rangle$ **have** $\langle c\ \#_{n'}\ inf\ llist\ t \rangle \leq \langle c\ \#_{n''}\ inf\ llist\ t \rangle$

using *nAct-mono-lNact* **by** *simp*

moreover from $\langle n' \geq \langle c \Leftarrow t \rangle_n \rangle$ **have** $\langle c\ \#_n\ inf\ llist\ t \rangle \leq \langle c\ \#_{n'}\ inf\ llist\ t \rangle$

using *nAct-mono-lNact* **by** *simp*

moreover have $\langle c\ \#_{n'}\ inf\ llist\ t \rangle \neq \infty$ **by** *simp*

ultimately show *?thesis* **by** *simp*

qed

moreover have $\exists i' \geq n. \|c\|_t i'$

proof –

from $\langle \exists i' \geq n'. \|c\|_t i' \rangle$ **obtain** i' **where** $i' \geq n'$ **and** $\|c\|_t i'$ **by** *blast*

with $\langle n' \geq \langle c \Leftarrow t \rangle_n \rangle$ **have** $i' \geq n$ **using** *lNactGe\ le-trans* **by** *blast*

with $\langle \|c\|_t i' \rangle$ **show** *?thesis* **by** *blast*

qed

ultimately have $eval\ c\ t\ t'\ n\ (\lambda t\ n. \exists n' \geq n. \gamma\ t\ n')$

using *validCI-act* [**where** $\gamma = (\lambda t\ n. \exists n' \geq n. \gamma\ t\ n')$] **by** *blast*

thus *?thesis* using *evt-def* by *simp*
 next
 assume $\neg(\exists i' \geq n'. \|c\|_{t i'})$
 with $\langle \exists i \geq n. \|c\|_{t i} \rangle$ have $n' \geq \langle c \wedge t \rangle$ using *lActive-less* by *auto*
 hence $c \downarrow_t(n') \geq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1$ using *cnf2bhv-ge-llength* by *simp*
 moreover have $\text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1 \geq \text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle)$
 proof –
 from $\langle \exists i \geq n. \|c\|_{t i} \rangle$ have $\text{llength}(\pi_c(\text{inf-llist } t)) \geq \text{eSuc}(\langle c \#_n \text{inf-llist } t \rangle)$
 using *nAct-llength-proj* by *simp*
 moreover from $\langle \neg(\exists i' \geq n'. \|c\|_{t i'}) \rangle$ have *lfinite* $(\pi_c(\text{inf-llist } t))$
 using *proj-finite2[of inf-llist t]* by *simp*
 hence $\text{llength}(\pi_c(\text{inf-llist } t)) \neq \infty$ using *llength-eq-infty-conv-lfinite* by *auto*
 ultimately have $\text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) \geq \text{the-enat}(\text{eSuc}(\langle c \#_n \text{inf-llist } t \rangle))$
 by *simp*
 moreover have $\langle c \#_n \text{inf-llist } t \rangle \neq \infty$ by *simp*
 ultimately have $\text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) \geq \text{Suc}(\text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle))$
 using *the-enat-eSuc* by *simp*
 thus *?thesis* by *simp*
 qed
 ultimately have $c \downarrow_t(n') \geq \text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle)$ by *simp*
 moreover from $\langle \neg(\exists i' \geq n'. \|c\|_{t i'}) \rangle$ have *eval c t t' n' γ* using *assms(4)* by *simp*
 with $\langle \exists i \geq n. \|c\|_{t i} \rangle \langle \neg(\exists i' \geq n'. \|c\|_{t i'}) \rangle$
 have $\gamma(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')))(c \downarrow_t(n'))$ using *validCE-cont* by *blast*
 ultimately have *eval c t t' n* $(\lambda t n. \exists n' \geq n. \gamma t n')$
 using $\langle \exists i \geq n. \|c\|_{t i} \rangle$ *validCI-act[where $\gamma = (\lambda t n. \exists n' \geq n. \gamma t n')$]* by *blast*
 thus *?thesis* using *evt-def* by *simp*
 qed
 lemma *evtIN[intro]*:
 fixes *c*::*'id*
 and *t*::*nat* \Rightarrow *cnf*
 and *t'*::*nat* \Rightarrow *'cmp*
 and *n*::*nat*
 and *n'*::*nat*
 assumes $\neg(\exists i \geq n. \|c\|_{t i})$
 and $n' \geq n$
 and *eval c t t' n' γ*
 shows *eval c t t' n* $(\Diamond_b(\gamma))$
 proof *cases*
 assume $\exists i. \|c\|_{t i}$
 moreover from *assms(1)* *assms(2)* have $\neg(\exists i' \geq n'. \|c\|_{t i'})$ by *simp*
 ultimately have $\gamma(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')))(c \downarrow_t(n'))$
 using *validCE-cont[of c t n' t' γ]* *eval c t t' n' γ* by *blast*
 moreover from $\langle n' \geq n \rangle$ have $c \downarrow_t(n') \geq c \downarrow_t(n)$ using *cnf2bhv-mono* by *simp*
 ultimately have *eval c t t' n* $(\lambda t n. \exists n' \geq n. \gamma t n')$
 using *validCI-cont[where $\gamma = (\lambda t n. \exists n' \geq n. \gamma t n')$]* $\langle \exists i. \|c\|_{t i} \rangle \langle \neg(\exists i \geq n. \|c\|_{t i}) \rangle$ by *blast*
 thus *?thesis* using *evt-def* by *simp*
 next
 assume $\neg(\exists i. \|c\|_{t i})$
 with *assms* have $\gamma(\text{lnth}(\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) n'$ using *validCE-not-act* by *blast*
 with $\langle \neg(\exists i. \|c\|_{t i}) \rangle$ have *eval c t t' n* $(\lambda t n. \exists n' \geq n. \gamma t n')$
 using *validCI-not-act[where $\gamma = \lambda t n. \exists n' \geq n. \gamma t n'$]* $\langle n' \geq n \rangle$ by *blast*
 thus *?thesis* using *evt-def* by *simp*
 qed

lemma *evtEA*[*elim*]:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

assumes $\exists i \geq n. \|c\|_t i$

and $eval\ c\ t\ t'\ n\ (\Diamond_b(\gamma))$

shows $\exists n' \geq \langle c \rightarrow t \rangle_n.$

$(\exists i \geq n'. \|c\|_t i \wedge (\forall n'' \geq \langle c \leftarrow t \rangle_{n'}. n'' \leq \langle c \rightarrow t \rangle_{n'} \longrightarrow eval\ c\ t\ t'\ n''\ \gamma)) \vee$

$(\neg(\exists i \geq n'. \|c\|_t i) \wedge eval\ c\ t\ t'\ n'\ \gamma)$

proof –

from $\langle eval\ c\ t\ t'\ n\ (\Diamond_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n. \exists n' \geq n. \gamma\ t\ n')$ using *evt-def* by *simp*

with $\langle \exists i \geq n. \|c\|_t i \rangle$

have $\exists n' \geq the-enat\ \langle c\ \#_{enat}\ n\ inf-llist\ t \rangle. \gamma\ (lnth\ (\pi_c\ inf-llist\ t\ @_l\ inf-llist\ t'))\ n'$

using *validCE-act*[**where** $\gamma = \lambda t\ n. \exists n' \geq n. \gamma\ t\ n'$] **by** *blast*

then obtain x where $x \geq the-enat\ (\langle c\ \#_n\ inf-llist\ t \rangle)$ and

$\gamma\ (lnth\ ((\pi_c\ (inf-llist\ t))\ @_l\ (inf-llist\ t'))) x$ by *auto*

thus *?thesis*

proof (*cases*)

assume $x \geq llength\ (\pi_c\ (inf-llist\ t))$

moreover from $\langle x \geq llength\ (\pi_c\ (inf-llist\ t)) \rangle$ have $llength\ (\pi_c\ (inf-llist\ t)) \neq \infty$

by (*metis infinity-ileE*)

moreover from $\langle \exists i \geq n. \|c\|_t i \rangle$ have $llength\ (\pi_c\ (inf-llist\ t)) \geq 1$

using *proj-one*[*of inf-llist t*] **by** *auto*

ultimately have $the-enat\ (llength\ (\pi_c\ (inf-llist\ t))) - 1 < x$

by (*metis One-nat-def Suc-ile-eq antisym-conv2 diff-Suc-less enat-ord-simps*(2)
enat-the-enat less-imp-diff-less one-enat-def)

hence $x = c \downarrow_t (c \uparrow_t(x))$ using *cnf2bhv-bhv2cnf* **by** *simp*

with $\langle \gamma\ (lnth\ ((\pi_c\ (inf-llist\ t))\ @_l\ (inf-llist\ t'))) x \rangle$

have $\gamma\ (lnth\ ((\pi_c\ (inf-llist\ t))\ @_l\ (inf-llist\ t'))) (c \downarrow_t (c \uparrow_t(x)))$ **by** *simp*

moreover have $\neg(\exists i \geq c \uparrow_t(x). \|c\|_t i)$

proof –

from $\langle x \geq llength\ (\pi_c\ (inf-llist\ t)) \rangle$ have *lfinite* $(\pi_c\ (inf-llist\ t))$

using *llength-geq-enat-lfiniteD*[*of* $\pi_c\ (inf-llist\ t)\ x$] **by** *simp*

then obtain z where $\forall n'' > z. \neg \|c\|_t n''$ using *proj-finite-bound* **by** *blast*

moreover from $\langle the-enat\ (llength\ (\pi_c\ (inf-llist\ t))) - 1 < x \rangle$ have $\langle c \wedge t \rangle < c \uparrow_t(x)$

using *bhv2cnf-greater-lActive* **by** *simp*

ultimately show *?thesis* using *lActive-greater-active-all* **by** *simp*

qed

ultimately have $eval\ c\ t\ t'\ (c \uparrow_t(x))\ \gamma$

using $\langle \exists i \geq n. \|c\|_t i \rangle$ *validCI-cont*[*of* $c\ t\ c \uparrow_t(x)$] **by** *blast*

moreover have $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$

proof –

from $\langle x \geq llength\ (\pi_c\ (inf-llist\ t)) \rangle$ have *lfinite* $(\pi_c\ (inf-llist\ t))$

using *llength-geq-enat-lfiniteD*[*of* $\pi_c\ (inf-llist\ t)\ x$] **by** *simp*

then obtain z where $\forall n'' > z. \neg \|c\|_t n''$ using *proj-finite-bound* **by** *blast*

moreover from $\langle \exists i \geq n. \|c\|_t i \rangle$ have $\|c\|_t \langle c \rightarrow t \rangle_n$ using *nextActI* **by** *simp*

ultimately have $\langle c \wedge t \rangle \geq \langle c \rightarrow t \rangle_n$ using *lActive-greatest* **by** *fastforce*

moreover have $c \uparrow_t(x) \geq \langle c \wedge t \rangle$ **by** *simp*

ultimately show $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$ **by** *arith*

qed

ultimately show *?thesis* using $\langle \neg(\exists i \geq c \uparrow_t(x). \|c\|_t i) \rangle$ **by** *blast*

next

assume $\neg(x \geq llength\ (\pi_c\ (inf-llist\ t)))$

hence $x < llength\ (\pi_c\ (inf-llist\ t))$ **by** *simp*

then obtain $n'::nat$ where $x=\langle c \#_{n'} inf\text{-}llist\ t \rangle$ using $nAct\text{-}exists$ by *blast*
with $\langle enat\ x < llength\ (\pi_c(inf\text{-}llist\ t)) \rangle$ have $\exists i \geq n'. \|c\|_t\ i$ using $nAct\text{-}less\text{-}length\text{-}active$ by *force*
then obtain i where $i \geq n'$ and $\|c\|_t\ i$ and $\neg (\exists k \geq n'. k < i \wedge \|c\|_t\ k)$ using $nact\text{-}exists$ by *blast*
moreover have $(\forall n'' \geq \langle c \Leftarrow t \rangle_i. n'' \leq \langle c \rightarrow t \rangle_i \longrightarrow eval\ c\ t\ t'\ n''\ \gamma)$

proof

fix n'' show $\langle c \Leftarrow t \rangle_i \leq n'' \longrightarrow n'' \leq \langle c \rightarrow t \rangle_i \longrightarrow eval\ c\ t\ t'\ n''\ \gamma$

proof(rule *HOL.impI*[*OF HOL.impI*])

assume $\langle c \Leftarrow t \rangle_i \leq n''$ and $n'' \leq \langle c \rightarrow t \rangle_i$

hence $the\text{-}enat\ (\langle c \#_{enat\ i}\ inf\text{-}llist\ t \rangle) = the\text{-}enat\ (\langle c \#_{enat\ n''}\ inf\text{-}llist\ t \rangle)$

using $nAct\text{-}same$ by *simp*

moreover from $\langle \|c\|_t\ i \rangle$ have $\|c\|_t\ \langle c \rightarrow t \rangle_i$ using $nxtActI$ by *auto*

with $\langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$ have $\exists i \geq n'''. \|c\|_t\ i$ using $dual\text{-}order.\text{strict}\text{-}implies\text{-}order$ by *auto*

moreover have $\gamma\ (lnth\ ((\pi_c(inf\text{-}llist\ t))\ @_l\ (inf\text{-}llist\ t')))\ (the\text{-}enat\ (\langle c \#_{enat\ i}\ inf\text{-}llist\ t \rangle))$

proof –

have $enat\ i - 1 < llength\ (inf\text{-}llist\ t)$ by (*simp add: one-enat-def*)

with $\langle x = \langle c \#_{n'}\ inf\text{-}llist\ t \rangle \langle i \geq n' \rangle \langle \neg (\exists k \geq n'. k < i \wedge \|c\|_t\ k) \rangle$ have $x = \langle c \#_i\ inf\text{-}llist\ t \rangle$

using $one\text{-}enat\text{-}def\ nAct\text{-}not\text{-}active\text{-}same$ by *simp*

moreover have $\langle c \#_i\ inf\text{-}llist\ t \rangle \neq \infty$ by *simp*

ultimately have $x = the\text{-}enat(\langle c \#_i\ inf\text{-}llist\ t \rangle)$ by *fastforce*

thus $?thesis$ using $\langle \gamma\ (lnth\ ((\pi_c(inf\text{-}llist\ t))\ @_l\ (inf\text{-}llist\ t')))\ x \rangle$ by *blast*

qed

with $\langle the\text{-}enat\ (\langle c \#_{enat\ i}\ inf\text{-}llist\ t \rangle) = the\text{-}enat\ (\langle c \#_{enat\ n''}\ inf\text{-}llist\ t \rangle) \rangle$ have

$\gamma\ (lnth\ ((\pi_c(inf\text{-}llist\ t))\ @_l\ (inf\text{-}llist\ t')))\ (the\text{-}enat\ (\langle c \#_{enat\ n''}\ inf\text{-}llist\ t \rangle))$ by *simp*

ultimately show $eval\ c\ t\ t'\ n''\ \gamma$ using $validCI\text{-}act$ by *blast*

qed

qed

moreover have $i \geq \langle c \rightarrow t \rangle_n$

proof –

have $enat\ i - 1 < llength\ (inf\text{-}llist\ t)$ by (*simp add: one-enat-def*)

with $\langle x = \langle c \#_{n'}\ inf\text{-}llist\ t \rangle \langle i \geq n' \rangle \langle \neg (\exists k \geq n'. k < i \wedge \|c\|_t\ k) \rangle$ have $x = \langle c \#_i\ inf\text{-}llist\ t \rangle$

using $one\text{-}enat\text{-}def\ nAct\text{-}not\text{-}active\text{-}same$ by *simp*

moreover have $\langle c \#_i\ inf\text{-}llist\ t \rangle \neq \infty$ by *simp*

ultimately have $x = the\text{-}enat(\langle c \#_i\ inf\text{-}llist\ t \rangle)$ by *fastforce*

with $\langle x \geq the\text{-}enat\ (\langle c \#_n\ inf\text{-}llist\ t \rangle) \rangle$

have $the\text{-}enat\ (\langle c \#_i\ inf\text{-}llist\ t \rangle) \geq the\text{-}enat\ (\langle c \#_n\ inf\text{-}llist\ t \rangle)$ by *simp*

with $\langle \|c\|_t\ i \rangle$ show $?thesis$ using $active\text{-}geq\text{-}nxtAct$ by *simp*

qed

ultimately show $?thesis$ using $\langle \|c\|_t\ i \rangle$ by *auto*

qed

qed

lemma $evtEN[elim]$:

fixes $c::'id$

and $t::nat \Rightarrow cnf$

and $t'::nat \Rightarrow 'cmp$

and $n::nat$

and $n'::nat$

assumes $\neg (\exists i \geq n. \|c\|_t\ i)$

and $eval\ c\ t\ t'\ n\ (\Diamond_b(\gamma))$

shows $\exists n' \geq n. eval\ c\ t\ t'\ n'\ \gamma$

proof *cases*

assume $\exists i. \|c\|_t\ i$

moreover from $\langle eval\ c\ t\ t'\ n\ (\Diamond_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n. \exists n' \geq n. \gamma\ t\ n')$ using $evt\text{-}def$ by *simp*

ultimately have $\exists n' \geq c \downarrow t n. \gamma\ (lnth\ (\pi_c\ inf\text{-}llist\ t\ @_l\ inf\text{-}llist\ t'))\ n'$

using $validCE\text{-}cont$ [where $\gamma = (\lambda t\ n. \exists n' \geq n. \gamma\ t\ n')$] $\langle \neg (\exists i \geq n. \|c\|_t\ i) \rangle$ by *blast*

then obtain x where $x \geq c \downarrow_t(n)$ and $\gamma (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) x$ by *auto*
 moreover have *the-enat* $(\text{llength } (\pi_c(\text{inf-llist } t))) - 1 < x$

proof –

have $\langle c \wedge t \rangle < n$

proof (*rule ccontr*)

assume $\neg \langle c \wedge t \rangle < n$

hence $\langle c \wedge t \rangle \geq n$ by *simp*

moreover from $\langle \exists i. \|c\|_t i \rangle \langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ have $\|c\|_t \langle c \wedge t \rangle$

using *lActive-active less-or-eq-imp-le* by *blast*

ultimately show *False* using $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ by *simp*

qed

hence *the-enat* $(\text{llength } (\pi_c(\text{inf-llist } t))) - 1 < c \downarrow_t(n)$ using *cnf2bhv-greater-llength* by *simp*

with $\langle x \geq c \downarrow_t(n) \rangle$ show *?thesis* by *simp*

qed

hence $x = c \downarrow_t(c \uparrow_t(x))$ using *cnf2bhv-bhv2cnf* by *simp*

ultimately have $\gamma (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t(c \uparrow_t(x)))$ by *simp*

moreover from $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ have $\neg (\exists i \geq c \uparrow_t(x). \|c\|_t i)$

proof –

from $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ have *lfinite* $(\pi_c(\text{inf-llist } t))$ using *proj-finite2* by *simp*

then obtain z where $\forall n'' > z. \neg \|c\|_t n''$ using *proj-finite-bound* by *blast*

moreover from $\langle \text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))) - 1 < x \rangle$ have $\langle c \wedge t \rangle < c \uparrow_t(x)$

using *bhv2cnf-greater-lActive* by *simp*

ultimately show *?thesis* using *lActive-greater-active-all* by *simp*

qed

ultimately have *eval* $c t t' (c \uparrow_t x) \gamma$

using *validCI-cont*[of $c t c \uparrow_t(x) \gamma$] $\langle \exists i. \|c\|_t i \rangle$ by *blast*

moreover from $\langle \exists i. \|c\|_t i \rangle \langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ have $\langle c \wedge t \rangle \leq n$ using *lActive-less*[of $c t - n$] by *auto*

with $\langle x \geq c \downarrow_t(n) \rangle$ have $n \leq c \uparrow_t(x)$ using *p2c-mono-c2p* by *blast*

ultimately show *?thesis* by *auto*

next

assume $\neg (\exists i. \|c\|_t i)$

moreover from $\langle \text{eval } c t t' n (\diamond_b(\gamma)) \rangle$ have *eval* $c t t' n (\lambda t n. \exists n' \geq n. \gamma t n')$ using *evt-def* by *simp*

ultimately obtain n' where $n' \geq n$ and $\gamma (\text{lnth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) n'$

using $\langle \neg (\exists i. \|c\|_t i) \rangle$ *validCE-not-act*[where $\gamma = \lambda t n. \exists n' \geq n. \gamma t n'$] by *blast*

with $\langle \neg (\exists i. \|c\|_t i) \rangle$ show *?thesis* using *validCI-not-act*[of $c t \gamma t' n'$] by *blast*

qed

2.4.11 Globally Operator

definition *glob* :: $(\text{'cmp } bta) \Rightarrow (\text{'cmp } bta) (\square_b(-) \ 22)$

where $\square_b(\gamma) \equiv \lambda t n. \forall n' \geq n. \gamma t n'$

lemma *globIA*[*intro*]:

fixes $c :: \text{'id}$

and $t :: \text{nat} \Rightarrow \text{cnf}$

and $t' :: \text{nat} \Rightarrow \text{'cmp}$

and $n :: \text{nat}$

assumes $\exists i \geq n. \|c\|_t i$

and $\bigwedge n'. [\exists i \geq n'. \|c\|_t i; n' \geq \langle c \rightarrow t \rangle_n] \Longrightarrow \exists n'' \geq \langle c \leftarrow t \rangle_{n'}. n'' \leq \langle c \rightarrow t \rangle_{n'} \wedge \text{eval } c t t' n'' \gamma$

and $\bigwedge n'. [\neg (\exists i \geq n'. \|c\|_t i); n' \geq \langle c \rightarrow t \rangle_n] \Longrightarrow \text{eval } c t t' n' \gamma$

shows *eval* $c t t' n (\square_b(\gamma))$

proof –

have $\forall n' \geq \text{the-enat } \langle c \#_{\text{enat } n} \text{inf-llist } t \rangle. \gamma (\text{lnth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) n'$

proof

fix $x :: \text{nat}$ show

$x \geq \text{the-enat} (\langle c \#_n \text{ inf-llist } t \rangle) \longrightarrow \gamma (\text{lnth} (\pi_c \text{ inf-llist } t @_i \text{ inf-llist } t')) x$

proof

assume $x \geq \text{the-enat} (\langle c \#_n \text{ inf-llist } t \rangle)$

show $\gamma (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_i (\text{inf-llist } t'))) x$

proof (*cases*)

assume $(x \geq \text{llength} (\pi_c(\text{inf-llist } t)))$

hence $\text{lfinite} (\pi_c(\text{inf-llist } t))$

using $\text{llength-geq-enat-lfiniteD}$ [of $\pi_c(\text{inf-llist } t) x$] **by** *simp*

then obtain z **where** $\forall n'' > z. \neg \|c\|_{t n''}$ **using** *proj-finite-bound* **by** *blast*

moreover have $\|c\|_{t \langle c \rightarrow t \rangle_n}$ **by** (*simp add*: $\langle \exists i \geq n. \|c\|_{t i} \rangle \text{ nActI}$)

ultimately have $\langle c \wedge t \rangle \geq \langle c \rightarrow t \rangle_n$ **using** *LActive-greatest*[of $c t \langle c \rightarrow t \rangle_n$] **by** *blast*

moreover have $c \uparrow_t(x) \geq \langle c \wedge t \rangle$ **by** *simp*

ultimately have $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$ **by** *arith*

moreover have $\neg (\exists i' \geq c \uparrow_t(x). \|c\|_{t i'})$

proof –

from $\langle \text{lfinite} (\pi_c(\text{inf-llist } t)) \rangle \langle \exists i \geq n. \|c\|_{t i} \rangle$

have $c \uparrow_t(\text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t)))) = \text{Suc} (\langle c \wedge t \rangle)$

using *bhv2cnf-LActive* **by** *blast*

moreover from $\langle x \geq \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$ **have** $x \geq \text{the-enat}(\text{llength} (\pi_c(\text{inf-llist } t)))$

using *the-enat-mono* **by** *fastforce*

hence $c \uparrow_t(x) \geq c \uparrow_t(\text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t))))$

using *bhv2cnf-mono*[of *the-enat* ($\text{llength} (\pi_c(\text{inf-llist } t))$) x] **by** *simp*

ultimately have $c \uparrow_t(x) \geq \text{Suc} (\langle c \wedge t \rangle)$ **by** *simp*

hence $c \uparrow_t(x) > \langle c \wedge t \rangle$ **by** *simp*

with $\langle \forall n'' > z. \neg \|c\|_{t n''} \rangle$ **show** *?thesis* **using** *LActive-greater-active-all* **by** *simp*

qed

ultimately have $\text{eval } c t t' (c \uparrow_t(x)) \gamma$ **using** *assms(3)* **by** *simp*

hence $\gamma (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_i (\text{inf-llist } t'))) (c \downarrow_t(c \uparrow_t(x)))$

using *validCE-cont*[of $c t c \uparrow_t(x) t' \gamma$] $\langle \exists i \geq n. \|c\|_{t i} \rangle \langle \neg (\exists i' \geq c \uparrow_t(x). \|c\|_{t i'}) \rangle$ **by** *blast*

moreover from $\langle x \geq \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$

have $(\text{enat } x \geq \text{llength} (\pi_c(\text{inf-llist } t)))$ **by** *auto*

with $\langle \text{lfinite} (\pi_c(\text{inf-llist } t)) \rangle$ **have** $\text{llength} (\pi_c(\text{inf-llist } t)) \neq \infty$

using *llength-eq-infty-conv-lfinite* **by** *auto*

with $\langle x \geq \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$

have $\text{the-enat}(\text{llength} (\pi_c(\text{inf-llist } t))) - 1 \leq x$ **by** *auto*

ultimately show *?thesis* **using** *cnf2bhv-bhv2cnf*[of $c t x$] **by** *simp*

next

assume $\neg(x \geq \text{llength} (\pi_c(\text{inf-llist } t)))$

hence $x < \text{llength} (\pi_c(\text{inf-llist } t))$ **by** *simp*

then obtain $n' : \text{nat}$ **where** $x = \langle c \#_{n'} \text{ inf-llist } t \rangle$ **using** *nAct-exists* **by** *blast*

moreover from $\langle \text{enat } x < \text{llength} (\pi_c(\text{inf-llist } t)) \rangle \langle \text{enat } x = \langle c \#_{\text{enat } n'} \text{ inf-llist } t \rangle \rangle$

have $\exists i \geq n'. \|c\|_{t i}$ **using** *nAct-less-llength-active* **by** *force*

then obtain i **where** $i \geq n'$ **and** $\|c\|_{t i}$ **and** $\neg (\exists k \geq n'. k < i \wedge \|c\|_{t k})$

using *nact-exists* **by** *blast*

moreover have $\text{enat } i - 1 < \text{llength} (\text{inf-llist } t)$ **by** (*simp add*: *one-enat-def*)

ultimately have $x = \langle c \#_i \text{ inf-llist } t \rangle$ **using** *one-enat-def* *nAct-not-active-same* **by** *simp*

moreover have $\langle c \#_i \text{ inf-llist } t \rangle \neq \infty$ **by** *simp*

ultimately have $x = \text{the-enat}(\langle c \#_i \text{ inf-llist } t \rangle)$ **by** *fastforce*

from $\langle x \geq \text{the-enat} (\langle c \#_n \text{ inf-llist } t \rangle) \rangle \langle x = \text{the-enat} (\langle c \#_i \text{ inf-llist } t \rangle) \rangle$

have $\text{the-enat} (\langle c \#_i \text{ inf-llist } t \rangle) \geq \text{the-enat} (\langle c \#_n \text{ inf-llist } t \rangle)$ **by** *simp*

with $\langle \|c\|_{t i} \rangle$ **have** $i \geq \langle c \rightarrow t \rangle_n$ **using** *active-geq-nAct* **by** *simp*

moreover from $\langle x = \langle c \#_i \text{ inf-llist } t \rangle \rangle \langle x < \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$

have $\exists i'. i \leq \text{enat } i' \wedge \|c\|_{t i'}$ **using** *nAct-less-llength-active*[of $x c \text{ inf-llist } t i$] **by** *simp*

hence $\exists i' \geq i. \|c\|_{t i'}$ **by** *simp*

ultimately obtain n'' **where** $\text{eval } c t t' n'' \gamma$ **and** $n'' \geq \langle c \leftarrow t \rangle_i$ **and** $n'' \leq \langle c \rightarrow t \rangle_i$

using *assms*(2) by *blast*
 moreover have $\exists i' \geq n''. \|c\|_{t\ i'}$
 using $\langle \|c\|_{t\ i'} \langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$ *less-or-eq-imp-le nextAct-active* by *auto*
 ultimately have γ (*lnth* ($(\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')$)) (*the-enat* ($\langle c \#_{n''} \text{inf-llist } t \rangle$))
 using *validCE-act*[of $n''\ c\ t\ t'\ \gamma$] by *blast*
 moreover from $\langle n'' \geq \langle c \leftarrow t \rangle_i \rangle$ and $\langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$
 have *the-enat* ($\langle c \#_{n''} \text{inf-llist } t \rangle$) = *the-enat* ($\langle c \#_i \text{inf-llist } t \rangle$) using *nAct-same* by *simp*
 hence *the-enat* ($\langle c \#_{n''} \text{inf-llist } t \rangle$) = x by (*simp add*: $\langle x = \text{the-enat } \langle c \#_{\text{enat } i} \text{inf-llist } t \rangle \rangle$)
 ultimately have γ (*lnth* ($(\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')$)) (*the-enat* x) by *simp*
 thus ?*thesis* by *simp*
 qed
 qed
 qed
 with $\langle \exists i \geq n. \|c\|_{t\ i} \rangle$ have *eval* $c\ t\ t'\ n$ ($\lambda t\ n. \forall n' \geq n. \gamma\ t\ n'$)
 using *validCI-act*[of $n\ c\ t\ \lambda t\ n. \forall n' \geq n. \gamma\ t\ n'\ t'$] by *blast*
 thus ?*thesis* using *glob-def* by *simp*
 qed

lemma *globIN*[*intro*]:

fixes $c::'id$
 and $t::nat \Rightarrow cnf$
 and $t'::nat \Rightarrow 'cmp$
 and $n::nat$
 assumes $\neg(\exists i \geq n. \|c\|_{t\ i})$
 and $\bigwedge n'. n' \geq n \implies \text{eval } c\ t\ t'\ n'\ \gamma$
 shows *eval* $c\ t\ t'\ n$ ($\square_b(\gamma)$)

proof *cases*

assume $\exists i. \|c\|_{t\ i}$
 from $\langle \neg(\exists i \geq n. \|c\|_{t\ i}) \rangle$ have *lfinite* ($\pi_c(\text{inf-llist } t)$) using *proj-finite2* by *simp*
 then obtain z where $\forall n'' > z. \neg \|c\|_{t\ n''}$ using *proj-finite-bound* by *blast*

have $\forall x::nat \geq c \downarrow_t(n). \gamma$ (*lnth* ($\pi_c \text{inf-llist } t @_l \text{inf-llist } t'$)) x

proof

fix $x::nat$ show $(x \geq c \downarrow_t(n)) \longrightarrow \gamma$ (*lnth* ($\pi_c \text{inf-llist } t @_l \text{inf-llist } t'$)) x

proof

assume $x \geq c \downarrow_t(n)$
 moreover from $\langle \neg(\exists i \geq n. \|c\|_{t\ i}) \rangle$ have $\langle c \wedge t \rangle \leq n$ using $\langle \exists i. \|c\|_{t\ i} \rangle$ *lActive-less* by *auto*
 ultimately have $c \uparrow_t(x) \geq n$ using *p2c-mono-c2p* by *simp*
 with *assms* have *eval* $c\ t\ t'$ ($c \uparrow_t(x)$) γ by *simp*
 moreover have $\neg(\exists i' \geq c \uparrow_t(x). \|c\|_{t\ i'})$

proof –

from $\langle \text{lfinite } (\pi_c(\text{inf-llist } t)) \rangle$ $\langle \exists i. \|c\|_{t\ i} \rangle$
 have $c \uparrow_t(\text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t)))) = \text{Suc } (\langle c \wedge t \rangle)$
 using *bhv2cnf-lActive* by *blast*

moreover from $\langle \neg(\exists i \geq n. \|c\|_{t\ i}) \rangle$ have $n > \langle c \wedge t \rangle$

by (*meson* $\langle \exists i. \|c\|_{t\ i} \rangle$ *lActive-active leI le-eq-less-or-eq*)

hence $n \geq \text{Suc } (\langle c \wedge t \rangle)$ by *simp*

with $\langle n \geq \text{Suc } (\langle c \wedge t \rangle) \rangle$ $\langle c \uparrow_t(x) \geq n \rangle$ have $c \uparrow_t(x) \geq \text{Suc } (\langle c \wedge t \rangle)$ by *simp*

hence $c \uparrow_t(x) > \langle c \wedge t \rangle$ by *simp*

with $\langle \forall n'' > z. \neg \|c\|_{t\ n''} \rangle$ show ?*thesis* using *lActive-greater-active-all* by *simp*

qed

ultimately have γ (*lnth* ($(\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t')$)) ($c \downarrow_t(c \uparrow_t(x))$)

using *validCE-cont*[of $c\ t\ c \uparrow_t(x)\ t'\ \gamma$] $\langle \exists i. \|c\|_{t\ i} \rangle$ by *blast*

moreover have $x \geq \text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))) - 1$

using $\langle c \downarrow_t(n) \leq x \rangle$ *cnf2bhv-def* by *auto*

ultimately show γ ($\text{lnth } ((\pi_c(\text{inf-list } t)) @_l (\text{inf-list } t'))$) x
using $\text{cnf}^2\text{bhv-bhv}^2\text{cnf}$ **by** simp
qed
qed
with $\langle \exists i. \|c\|_t i \rangle \langle \neg(\exists i \geq n. \|c\|_t i) \rangle$ **have** $\text{eval } c \ t \ t' \ n$ ($\lambda t \ n. \forall n' \geq n. \gamma \ t \ n'$)
using validCI-cont [$\text{of } c \ t \ n \ \lambda t \ n. \forall n' \geq n. \gamma \ t \ n' \ t'$] **by** simp
thus $?thesis$ **using** glob-def **by** simp
next
assume $\neg(\exists i. \|c\|_t i)$
with assms **have** $\forall n' \geq n. \gamma$ ($\text{lnth } (\pi_c \text{inf-list } t @_l \text{inf-list } t')$) n' **using** validCE-not-act **by** blast
with $\langle \neg(\exists i. \|c\|_t i) \rangle$ **have** $\text{eval } c \ t \ t' \ n$ ($\lambda t \ n. \forall n' \geq n. \gamma \ t \ n'$)
using validCI-not-act [**where** $\gamma = \lambda t \ n. \forall n' \geq n. \gamma \ t \ n'$] **by** blast
thus $?thesis$ **using** glob-def **by** simp
qed

lemma globEA [elim]:
fixes $c::'id$
and $t::\text{nat} \Rightarrow \text{cnf}$
and $t'::\text{nat} \Rightarrow 'cmp$
and $n::\text{nat}$
and $n'::\text{nat}$
assumes $\exists i \geq n. \|c\|_t i$
and $\text{eval } c \ t \ t' \ n$ ($\square_b(\gamma)$)
and $n' \geq \langle c \leftarrow t \rangle_n$
shows $\text{eval } c \ t \ t' \ n' \ \gamma$
proof (cases)
assume $\exists i \geq n'. \|c\|_t i$
with $\langle n' \geq \langle c \leftarrow t \rangle_n \rangle$ **have** $\text{the-enat } (\langle c \#_{n'} \text{inf-list } t \rangle) \geq \text{the-enat } (\langle c \#_n \text{inf-list } t \rangle)$
using $n\text{Act-mono-lNact}$ $\langle \exists i \geq n. \|c\|_t i \rangle$ **by** simp
moreover from $\langle \text{eval } c \ t \ t' \ n$ ($\square_b(\gamma)$) **have** $\text{eval } c \ t \ t' \ n$ ($\lambda t \ n. \forall n' \geq n. \gamma \ t \ n'$)
using glob-def **by** simp
hence $\forall x \geq \text{the-enat } \langle c \#_{\text{enat } n} \text{inf-list } t \rangle. \gamma$ ($\text{lnth } (\pi_c \text{inf-list } t @_l \text{inf-list } t')$) x
using validCE-act $\langle \exists i \geq n. \|c\|_t i \rangle$ **by** blast
ultimately have γ ($\text{lnth } ((\pi_c(\text{inf-list } t)) @_l (\text{inf-list } t'))$) ($\text{the-enat } (\langle c \#_{n'} \text{inf-list } t \rangle)$) **by** simp
with $\langle \exists i \geq n'. \|c\|_t i \rangle$ **show** $?thesis$ **using** validCI-act **by** blast
next
assume $\neg(\exists i \geq n'. \|c\|_t i)$
from $\langle \text{eval } c \ t \ t' \ n$ ($\square_b(\gamma)$) **have** $\text{eval } c \ t \ t' \ n$ ($\lambda t \ n. \forall n' \geq n. \gamma \ t \ n'$) **using** glob-def **by** simp
hence $\forall x \geq \text{the-enat } \langle c \#_{\text{enat } n} \text{inf-list } t \rangle. \gamma$ ($\text{lnth } (\pi_c \text{inf-list } t @_l \text{inf-list } t')$) x
using validCE-act $\langle \exists i \geq n. \|c\|_t i \rangle$ **by** blast
moreover have $c \downarrow_t(n') \geq \text{the-enat } (\langle c \#_n \text{inf-list } t \rangle)$
proof –
have $\langle c \#_n \text{inf-list } t \rangle \leq \text{length } (\pi_c(\text{inf-list } t))$ **using** $n\text{Act-le-proj}$ **by** metis
moreover from $\langle \neg(\exists i \geq n'. \|c\|_t i) \rangle$ **have** $\text{length } (\pi_c(\text{inf-list } t)) \neq \infty$
by ($\text{metis } \text{length-eq-inf-conv-lfinite } \text{lnth-inf-list } \text{proj-finite2}$)
ultimately have $\text{the-enat}(\langle c \#_n \text{inf-list } t \rangle) \leq \text{the-enat}(\text{length } (\pi_c(\text{inf-list } t)))$ **by** simp
moreover from $\langle \exists i \geq n. \|c\|_t i \rangle \langle \neg(\exists i \geq n'. \|c\|_t i) \rangle$ **have** $n' > \langle c \wedge t \rangle$
using $l\text{Active-active}$ **by** ($\text{meson } \text{leI } \text{le-eq-less-or-eq}$)
hence $c \downarrow_t(n') > \text{the-enat } (\text{length } (\pi_c(\text{inf-list } t))) - 1$ **using** $\text{cnf}^2\text{bhv-greater-length}$ **by** simp
ultimately show $?thesis$ **by** simp
qed
ultimately have γ ($\text{lnth } ((\pi_c(\text{inf-list } t)) @_l (\text{inf-list } t'))$) ($c \downarrow_t(n')$) **by** simp
with $\langle \exists i \geq n. \|c\|_t i \rangle \langle \neg(\exists i \geq n'. \|c\|_t i) \rangle$ **show** $?thesis$ **using** validCI-cont **by** blast
qed

lemma *globEANow*:

fixes $c\ t\ t'\ n\ i\ \gamma$
 assumes $n \leq i$
 and $\|c\|_t\ i$
 and $eval\ c\ t\ t'\ n\ (\Box_b\ \gamma)$
 shows $eval\ c\ t\ t'\ i\ \gamma$

proof –

from $\langle \|c\|_t\ i \rangle \langle n \leq i \rangle$ have $\exists i \geq n. \|c\|_t\ i$ by *auto*
 moreover from $\langle n \leq i \rangle$ have $\langle c \Leftarrow t \rangle_n \leq i$ using *dual-order.trans lNactLe* by *blast*
 ultimately show *?thesis* using *globEA[of n c t t' \gamma i] \langle eval\ c\ t\ t'\ n\ (\Box_b\ \gamma) \rangle* by *simp*
 qed

lemma *globEN[elim]*:

fixes $c::'id$
 and $t::nat \Rightarrow cnf$
 and $t'::nat \Rightarrow 'cmp$
 and $n::nat$
 and $n'::nat$
 assumes $\neg(\exists i \geq n. \|c\|_t\ i)$
 and $eval\ c\ t\ t'\ n\ (\Box_b(\gamma))$
 and $n' \geq n$
 shows $eval\ c\ t\ t'\ n'\ \gamma$

proof *cases*

assume $\exists i. \|c\|_t\ i$
 moreover from $\langle eval\ c\ t\ t'\ n\ (\Box_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n. \forall n' \geq n. \gamma\ t\ n')$
 using *glob-def* by *simp*
 ultimately have $\forall x \geq c \downarrow t\ n. \gamma\ (lnth\ (\pi_c\ inf\ llist\ t\ @_l\ inf\ llist\ t'))\ x$
 using *validCE-cont[of c t n t' \lambda t n. \forall n' \geq n. \gamma t n'] \langle \neg(\exists i \geq n. \|c\|_t\ i) \rangle* by *blast*
 moreover from $\langle n' \geq n \rangle$ have $c \downarrow t\ (n') \geq c \downarrow t\ n$ using *cnf2bhv-mono* by *simp*
 ultimately have $\gamma\ (lnth\ ((\pi_c\ (inf\ llist\ t))\ @_l\ (inf\ llist\ t'))\ (c \downarrow t\ (n')))$ by *simp*
 moreover from $\langle \neg(\exists i \geq n. \|c\|_t\ i) \rangle \langle n' \geq n \rangle$ have $\neg(\exists i \geq n'. \|c\|_t\ i)$ by *simp*
 ultimately show *?thesis* using *validCI-cont \langle \exists i. \|c\|_t\ i \rangle* by *blast*

next

assume $\neg(\exists i. \|c\|_t\ i)$
 moreover from $\langle eval\ c\ t\ t'\ n\ (\Box_b(\gamma)) \rangle$ have $eval\ c\ t\ t'\ n\ (\lambda t\ n. \forall n' \geq n. \gamma\ t\ n')$
 using *glob-def* by *simp*
 ultimately have $\forall n' \geq n. \gamma\ (lnth\ (\pi_c\ inf\ llist\ t\ @_l\ inf\ llist\ t'))\ n'$
 using $\langle \neg(\exists i. \|c\|_t\ i) \rangle$ *validCE-not-act[where \gamma = \lambda t n. \forall n' \geq n. \gamma t n']* by *blast*
 with $\langle \neg(\exists i. \|c\|_t\ i) \rangle \langle n' \geq n \rangle$ show *?thesis* using *validCI-not-act* by *blast*

qed

2.4.12 Until Operator

definition *until* :: $('cmp\ bta) \Rightarrow ('cmp\ bta) \Rightarrow ('cmp\ bta)$ (*infixl* $\mathfrak{U}_b\ 21$)
 where $\gamma' \mathfrak{U}_b\ \gamma \equiv \lambda t\ n. \exists n' \geq n. \gamma\ t\ n' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' t\ n')$

lemma *untilIA[intro]*:

fixes $c::'id$
 and $t::nat \Rightarrow cnf$
 and $t'::nat \Rightarrow 'cmp$
 and $n::nat$
 and $n'::nat$
 assumes $\exists i \geq n. \|c\|_t\ i$
 and $n' \geq \langle c \Leftarrow t \rangle_n$
 and $\llbracket \exists i \geq n'. \|c\|_t\ i \rrbracket \implies \exists n'' \geq \langle c \Leftarrow t \rangle_{n'}. n'' \leq \langle c \rightarrow t \rangle_{n'} \wedge eval\ c\ t\ t'\ n''\ \gamma \wedge$
 $(\forall n''' \geq \langle c \rightarrow t \rangle_n. n''' < \langle c \Leftarrow t \rangle_{n''})$

$\rightarrow (\exists n'''' \geq \langle c \Leftarrow t \rangle_{n''''}. n'''' \leq \langle c \rightarrow t \rangle_{n''''} \wedge \text{eval } c \ t \ t' \ n'''' \ \gamma')$
and $\llbracket \neg(\exists i \geq n'. \|c\|_t i) \rrbracket \Longrightarrow \text{eval } c \ t \ t' \ n' \ \gamma \wedge$
 $(\forall n'' \geq \langle c \rightarrow t \rangle_n. n'' < n')$
 $\rightarrow ((\exists i \geq n''. \|c\|_t i) \wedge (\exists n'''' \geq \langle c \Leftarrow t \rangle_{n''''}. n'''' \leq \langle c \rightarrow t \rangle_{n''''} \wedge \text{eval } c \ t \ t' \ n'''' \ \gamma')) \vee$
 $(\neg(\exists i \geq n''. \|c\|_t i) \wedge \text{eval } c \ t \ t' \ n' \ \gamma')$

shows $\text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{A}_b \ \gamma)$

proof cases

assume $\exists i' \geq n'. \|c\|_t i'$

with $\text{assms}(\beta)$ **obtain** n'' **where** $n'' \geq \langle c \Leftarrow t \rangle_{n'}$ **and** $n'' \leq \langle c \rightarrow t \rangle_{n'}$ **and** $\text{eval } c \ t \ t' \ n'' \ \gamma$ **and**

$a1: \forall n'''' \geq \langle c \rightarrow t \rangle_n. n'''' < \langle c \Leftarrow t \rangle_{n''}$

$\rightarrow (\exists n'''' \geq \langle c \Leftarrow t \rangle_{n''''}. n'''' \leq \langle c \rightarrow t \rangle_{n''''} \wedge \text{eval } c \ t \ t' \ n'''' \ \gamma')$ **by** *blast*

hence $\exists i' \geq n''. \|c\|_t i'$ **using** $\langle \exists i' \geq n'. \|c\|_t i' \rangle$ *nextActI* **by** *blast*

with $\langle \text{eval } c \ t \ t' \ n'' \ \gamma \rangle$ **have**

$\gamma \ (\text{lnth } ((\pi_c(\text{inf-llist } t)) \ @_l \ (\text{inf-llist } t'))) \ (\text{the-enat } (\langle c \ #_{n''} \ \text{inf-llist } t \rangle))$

using *validCE-act* **by** *blast*

moreover **have** $\text{the-enat } (\langle c \ #_n \ \text{inf-llist } t \rangle) \leq \text{the-enat } (\langle c \ #_{n''} \ \text{inf-llist } t \rangle)$

proof –

from $\langle \langle c \Leftarrow t \rangle_n \leq n' \rangle$ **have** $\langle c \ #_n \ \text{inf-llist } t \rangle \leq \langle c \ #_{n'} \ \text{inf-llist } t \rangle$

using *nAct-mono-lNact* **by** *simp*

moreover **from** $\langle \langle c \Leftarrow t \rangle_n \leq n'' \rangle$ **have** $\langle c \ #_{n'} \ \text{inf-llist } t \rangle \leq \langle c \ #_{n''} \ \text{inf-llist } t \rangle$

using *nAct-mono-lNact* **by** *simp*

ultimately **have** $\langle c \ #_n \ \text{inf-llist } t \rangle \leq \langle c \ #_{n''} \ \text{inf-llist } t \rangle$ **by** *simp*

moreover **have** $\langle c \ #_{n'} \ \text{inf-llist } t \rangle \neq \infty$ **by** *simp*

ultimately **show** *?thesis* **by** *simp*

qed

moreover **have** $\exists i' \geq n. \|c\|_t i'$

proof –

from $\langle \exists i' \geq n'. \|c\|_t i' \rangle$ **obtain** i' **where** $i' \geq n'$ **and** $\|c\|_t i'$ **by** *blast*

with $\langle n' \geq \langle c \Leftarrow t \rangle_n \rangle$ **have** $i' \geq n$ **using** *lNactGe le-trans* **by** *blast*

with $\langle \|c\|_t i' \rangle$ **show** *?thesis* **by** *blast*

qed

moreover **have** $\forall n' \geq \text{the-enat } \langle c \ #_n \ \text{inf-llist } t \rangle. n' < (\text{the-enat } \langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle)$

$\rightarrow \gamma' \ (\text{lnth } (\pi_c \ \text{inf-llist } t \ @_l \ \text{inf-llist } t')) \ n'$

proof

fix $x::\text{nat}$ **show** $x \geq \text{the-enat } (\langle c \ #_n \ \text{inf-llist } t \rangle)$

$\rightarrow x < (\text{the-enat } \langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle) \rightarrow \gamma' \ (\text{lnth } (\pi_c \ \text{inf-llist } t \ @_l \ \text{inf-llist } t')) \ x$

proof (*rule HOL.impI[OF HOL.impI]*)

assume $x \geq \text{the-enat } (\langle c \ #_n \ \text{inf-llist } t \rangle)$ **and** $x < (\text{the-enat } \langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle)$

moreover **have** $\text{the-enat } (\langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle) = \langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle$ **by** *simp*

ultimately **have** $x < \text{llength } (\pi_c(\text{inf-llist } t))$ **using** *nAct-le-proj*[of $c \ n'' \ \text{inf-llist } t$]

by (*metis enat-ord-simps(2) less-le-trans*)

hence $x < \text{llength } (\pi_c(\text{inf-llist } t))$ **by** *simp*

then **obtain** $n'::\text{nat}$ **where** $x = \langle c \ #_{n'} \ \text{inf-llist } t \rangle$ **using** *nAct-exists* **by** *blast*

moreover **from** $\langle \text{enat } x < \text{llength } (\pi_c(\text{inf-llist } t)) \rangle$ $\langle \text{enat } x = \langle c \ #_{\text{enat } n'} \ \text{inf-llist } t \rangle \rangle$

have $\exists i \geq n'. \|c\|_t i$ **using** *nAct-less-length-active* **by** *force*

then **obtain** i **where** $i \geq n'$ **and** $\|c\|_t i$ **and** $\neg(\exists k \geq n'. k < i \wedge \|c\|_t k)$ **using** *nAct-exists* **by** *blast*

moreover **have** $\text{enat } i - 1 < \text{llength } (\text{inf-llist } t)$ **by** (*simp add: one-enat-def*)

ultimately **have** $x = \langle c \ #_i \ \text{inf-llist } t \rangle$ **using** *one-enat-def* *nAct-not-active-same* **by** *simp*

moreover **have** $\langle c \ #_i \ \text{inf-llist } t \rangle \neq \infty$ **by** *simp*

ultimately **have** $x = \text{the-enat}(\langle c \ #_i \ \text{inf-llist } t \rangle)$ **by** *fastforce*

from $\langle x \geq \text{the-enat } (\langle c \ #_n \ \text{inf-llist } t \rangle) \rangle$ $\langle x = \text{the-enat}(\langle c \ #_i \ \text{inf-llist } t \rangle) \rangle$

have $\text{the-enat } (\langle c \ #_i \ \text{inf-llist } t \rangle) \geq \text{the-enat } (\langle c \ #_n \ \text{inf-llist } t \rangle)$ **by** *simp*

with $\langle \|c\|_t i \rangle$ **have** $i \geq \langle c \rightarrow t \rangle_n$ **using** *active-geq-nextAct* **by** *simp*

moreover **have** $i < \langle c \Leftarrow t \rangle_{n''}$

proof –

have $\text{the-enat } \langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle = \langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle$ **by** *simp*
with $\langle x < (\text{the-enat } \langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle) \rangle$ **and** $\langle x = \langle c \#_i \text{inf-llist } t \rangle \rangle$ **have**
 $\langle c \#_i \text{inf-llist } t \rangle < \langle c \#_{n''} \text{inf-llist } t \rangle$ **by** (*metis enat-ord-simps*(2))
hence $i < n''$ **using** *nAct-strict-mono-back*[of c i $\text{inf-llist } t$ n''] **by** *auto*
with $\langle \|c\|_t \rangle_i$ **show** *?thesis* **using** *lNact-notActive leI* **by** *blast*
qed
ultimately obtain n'' **where** $\text{eval } c \ t \ t' \ n'' \ \gamma'$ **and** $n'' \geq \langle c \Leftarrow t \rangle_i$ **and** $n'' \leq \langle c \rightarrow t \rangle_i$
using *a1* **by** *auto*
moreover have $\exists i' \geq n'' . \|c\|_{t \ i'}$
using $\langle \|c\|_t \rangle_i \ \langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$ *less-or-eq-imp-le nAct-active* **by** *auto*
ultimately have $\gamma' (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat } (\langle c \#_{n''} \text{inf-llist } t \rangle))$
using *validCE-act*[of n'' c t t' γ'] **by** *blast*
moreover from $\langle n'' \geq \langle c \Leftarrow t \rangle_i \rangle$ **and** $\langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$
have $\text{the-enat } (\langle c \#_{n''} \text{inf-llist } t \rangle) = \text{the-enat } (\langle c \#_i \text{inf-llist } t \rangle)$ **using** *nAct-same* **by** *simp*
hence $\text{the-enat } (\langle c \#_{n''} \text{inf-llist } t \rangle) = x$ **by** (*simp add*: $\langle x = \text{the-enat } \langle c \#_{\text{enat } i} \text{inf-llist } t \rangle \rangle$)
ultimately show $\gamma' (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) x$ **by** *simp*
qed
qed
ultimately have $\text{eval } c \ t \ t' \ n \ (\lambda \ t \ n . \exists n'' \geq n . \gamma \ t \ n'' \wedge (\forall n' \geq n . n' < n'' \longrightarrow \gamma' \ t \ n'))$
using *validCI-act*[**where** $\gamma = \lambda \ t \ n . \exists n'' \geq n . \gamma \ t \ n'' \wedge (\forall n' \geq n . n' < n'' \longrightarrow \gamma' \ t \ n')$] **by** *blast*
thus *?thesis* **using** *until-def* **by** *simp*
next
assume $\neg(\exists i' \geq n' . \|c\|_{t \ i'})$
with *assms*(4) **have** $\text{eval } c \ t \ t' \ n' \ \gamma$ **and** $a2: \forall n'' \geq \langle c \rightarrow t \rangle_n . n'' < n'$
 $\longrightarrow ((\exists i \geq n'' . \|c\|_{t \ i}) \wedge (\exists n''' \geq \langle c \Leftarrow t \rangle_{n''} . n''' \leq \langle c \rightarrow t \rangle_{n''} \wedge \text{eval } c \ t \ t' \ n''' \ \gamma')) \vee$
 $(\neg(\exists i \geq n'' . \|c\|_{t \ i}) \wedge \text{eval } c \ t \ t' \ n'' \ \gamma')$ **by** *auto*
with $\langle \neg(\exists i' \geq n' . \|c\|_{t \ i'}) \rangle$ $\langle \text{eval } c \ t \ t' \ n' \ \gamma \rangle$ $\langle \exists i \geq n . \|c\|_{t \ i} \rangle$ **have**
 $\gamma (\text{lnth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t (n'))$ **using** *validCE-cont* **by** *blast*
moreover have $c \downarrow_t (n') \geq \text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)$
proof –
from $\langle \exists i \geq n . \|c\|_{t \ i} \rangle$ $\langle \neg(\exists i' \geq n' . \|c\|_{t \ i'}) \rangle$ **have** $n' \geq \langle c \wedge t \rangle$ **using** *lActive-less* **by** *auto*
hence $c \downarrow_t (n') \geq \text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))) - 1$ **using** *cnf2bhv-ge-llength* **by** *simp*
moreover have $\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) - 1 \geq \text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle)$
proof –
from $\langle \exists i \geq n . \|c\|_{t \ i} \rangle$ **have** $\text{llength } (\pi_c(\text{inf-llist } t)) \geq e\text{Suc } (\langle c \#_n \text{inf-llist } t \rangle)$
using *nAct-llength-proj* **by** *simp*
moreover from $\langle \neg(\exists i' \geq n' . \|c\|_{t \ i'}) \rangle$ **have** *lfinite* $(\pi_c(\text{inf-llist } t))$
using *proj-finite2*[of $\text{inf-llist } t$] **by** *simp*
hence $\text{llength } (\pi_c(\text{inf-llist } t)) \neq \infty$ **using** *llength-eq-infnty-conv-lfinite* **by** *auto*
ultimately have $\text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))) \geq \text{the-enat}(e\text{Suc } (\langle c \#_n \text{inf-llist } t \rangle))$
by *simp*
moreover have $\langle c \#_n \text{inf-llist } t \rangle \neq \infty$ **by** *simp*
ultimately have $\text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))) \geq \text{Suc } (\text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle))$
using *the-enat-eSuc* **by** *simp*
thus *?thesis* **by** *simp*
qed
ultimately show *?thesis* **by** *simp*
qed
moreover have $\forall x \geq \text{the-enat } \langle c \#_n \text{inf-llist } t \rangle . x < (c \downarrow_t (n'))$
 $\longrightarrow \gamma' (\text{lnth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) x$
proof
fix $x::\text{nat}$ **show**
 $x \geq \text{the-enat } \langle c \#_n \text{inf-llist } t \rangle \longrightarrow x < (c \downarrow_t (n')) \longrightarrow \gamma' (\text{lnth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) x$
proof (*rule HOL.impI*[*OF HOL.impI*])
assume $x \geq \text{the-enat } \langle c \#_n \text{inf-llist } t \rangle$ **and** $x < (c \downarrow_t (n'))$

show $\gamma' (\text{lnth } ((\pi_c(\text{inf-llist } t)) \textcircled{=} (\text{inf-llist } t'))) x$
proof (*cases*)
assume $(x \geq \text{llength } (\pi_c(\text{inf-llist } t)))$
hence $\text{lfinite } (\pi_c(\text{inf-llist } t))$
using $\text{llength-geq-enat-lfiniteD}$ [of $\pi_c(\text{inf-llist } t)$ x] **by** *simp*
then obtain z **where** $\forall n'' > z. \neg \|c\|_{t \ n''}$ **using** *proj-finite-bound* **by** *blast*
moreover have $\|c\|_{t \ \langle c \rightarrow t \rangle_n}$ **by** (*simp add*: $\langle \exists i \geq n. \|c\|_{t \ i} \rangle \text{ nActI}$)
ultimately have $\langle c \wedge t \rangle \geq \langle c \rightarrow t \rangle_n$ **using** *LActive-greatest*[of $c \ t \ \langle c \rightarrow t \rangle_n$] **by** *blast*
moreover have $c \uparrow_t(x) \geq \langle c \wedge t \rangle$ **by** *simp*
ultimately have $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$ **by** *arith*
moreover have $\neg (\exists i' \geq c \uparrow_t(x). \|c\|_{t \ i'})$
proof –
from $\langle \text{lfinite } (\pi_c(\text{inf-llist } t)) \rangle \langle \exists i \geq n. \|c\|_{t \ i} \rangle$
have $c \uparrow_t(\text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t)))) = \text{Suc } (\langle c \wedge t \rangle)$
using *bhv2cnf-LActive* **by** *blast*
moreover from $\langle (x \geq \text{llength } (\pi_c(\text{inf-llist } t))) \rangle$ **have** $x \geq \text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t)))$
using *the-enat-mono* **by** *fastforce*
hence $c \uparrow_t(x) \geq c \uparrow_t(\text{the-enat } (\text{llength } (\pi_c(\text{inf-llist } t))))$
using *bhv2cnf-mono*[of *the-enat* ($\text{llength } (\pi_c(\text{inf-llist } t))$) x] **by** *simp*
ultimately have $c \uparrow_t(x) \geq \text{Suc } (\langle c \wedge t \rangle)$ **by** *simp*
hence $c \uparrow_t(x) > \langle c \wedge t \rangle$ **by** *simp*
with $\langle \forall n'' > z. \neg \|c\|_{t \ n''} \rangle$ **show** *?thesis* **using** *LActive-greater-active-all* **by** *simp*
qed
moreover have $c \uparrow_t x < n'$
proof –
from $\langle \text{lfinite } (\pi_c(\text{inf-llist } t)) \rangle$ **have** $\text{llength } (\pi_c \text{inf-llist } t) = \text{the-enat } (\text{llength } (\pi_c \text{inf-llist } t))$
by (*simp add*: *enat-the-enat llength-eq-infty-conv-lfinite*)
with $\langle x \geq \text{llength } (\pi_c(\text{inf-llist } t)) \rangle$ **have** $x \geq \text{the-enat } (\text{llength } (\pi_c \text{inf-llist } t))$
using *enat-ord-simps(1)* **by** *fastforce*
moreover from $\langle \exists i \geq n. \|c\|_{t \ i} \rangle$ **have** $\text{llength } (\pi_c \text{inf-llist } t) \geq 1$ **using** *proj-one* **by** *force*
ultimately have $\text{the-enat } (\text{llength } (\pi_c \text{inf-llist } t)) - 1 \leq x$ **by** *simp*
with $\langle x < (c \downarrow_t(n')) \rangle$ **show** *?thesis* **using** *c2p-mono-p2c-strict* **by** *simp*
qed
ultimately have $\text{eval } c \ t \ t' (c \uparrow_t(x)) \ \gamma'$ **using** *a2* **by** *blast*
hence $\gamma' (\text{lnth } ((\pi_c(\text{inf-llist } t)) \textcircled{=} (\text{inf-llist } t'))) (c \downarrow_t(c \uparrow_t(x)))$
using *validCE-cont*[of $c \ t \ c \uparrow_t(x) \ t' \ \gamma'$] $\langle \exists i \geq n. \|c\|_{t \ i} \rangle \langle \neg (\exists i' \geq c \uparrow_t(x). \|c\|_{t \ i'}) \rangle$ **by** *blast*
moreover from $\langle (x \geq \text{llength } (\pi_c(\text{inf-llist } t))) \rangle$
have $(\text{enat } x \geq \text{llength } (\pi_c(\text{inf-llist } t)))$ **by** *auto*
with $\langle \text{lfinite } (\pi_c(\text{inf-llist } t)) \rangle$ **have** $\text{llength } (\pi_c(\text{inf-llist } t)) \neq \infty$
using *llength-eq-infty-conv-lfinite* **by** *auto*
with $\langle (x \geq \text{llength } (\pi_c(\text{inf-llist } t))) \rangle$
have $\text{the-enat}(\text{llength } (\pi_c(\text{inf-llist } t))) - 1 \leq x$ **by** *auto*
ultimately show *?thesis* **using** *cnf2bhv-bhv2cnf*[of $c \ t \ x$] **by** *simp*
next
assume $\neg (x \geq \text{llength } (\pi_c(\text{inf-llist } t)))$
hence $x < \text{llength } (\pi_c(\text{inf-llist } t))$ **by** *simp*
then obtain $n'' :: \text{nat}$ **where** $x = \langle c \ \#_{n''} \text{inf-llist } t \rangle$ **using** *nAct-exists* **by** *blast*
moreover from $\langle \text{enat } x < \text{llength } (\pi_c(\text{inf-llist } t)) \rangle$ $\langle \text{enat } x = \langle c \ \#_{\text{enat } n''} \text{inf-llist } t \rangle \rangle$
have $\exists i \geq n'' . \|c\|_{t \ i}$ **using** *nAct-less-llength-active* **by** *force*
then obtain i **where** $i \geq n''$ **and** $\|c\|_{t \ i}$ **and** $\neg (\exists k \geq n'' . k < i \wedge \|c\|_{t \ k})$
using *nact-exists* **by** *blast*
moreover have $\text{enat } i - 1 < \text{llength } (\text{inf-llist } t)$ **by** (*simp add*: *one-enat-def*)
ultimately have $x = \langle c \ \#_i \text{inf-llist } t \rangle$ **using** *one-enat-def nAct-not-active-same* **by** *simp*
moreover have $\langle c \ \#_i \text{inf-llist } t \rangle \neq \infty$ **by** *simp*
ultimately have $x = \text{the-enat}(\langle c \ \#_i \text{inf-llist } t \rangle)$ **by** *fastforce*

from $\langle x \geq \text{the-enat} (\langle c \#_n \text{inf-llist } t \rangle) \rangle$ $\langle x = \text{the-enat} (\langle c \#_i \text{inf-llist } t \rangle) \rangle$
have $\text{the-enat} (\langle c \#_i \text{inf-llist } t \rangle) \geq \text{the-enat} (\langle c \#_n \text{inf-llist } t \rangle)$ **by simp**
with $\langle \|c\|_t \ i \rangle$ **have** $i \geq \langle c \rightarrow t \rangle_n$ **using active-geq-nxtAct by simp**
moreover from $\langle x = \langle c \#_i \text{inf-llist } t \rangle \rangle$ $\langle x < \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$
have $\exists i'. i \leq \text{enat } i' \wedge \|c\|_t \ i'$ **using nAct-less-llength-active[of x c inf-llist t i] by simp**
hence $\exists i' \geq i. \|c\|_t \ i'$ **by simp**
moreover have $i < n'$

proof –

from $\langle \exists i \geq n. \|c\|_t \ i \rangle$ $\langle \neg(\exists i' \geq n'. \|c\|_t \ i') \rangle$ **have** $n' \geq \langle c \wedge t \rangle$ **using lActive-less by auto**
hence $c \downarrow_t (n') \geq \text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t))) - 1$ **using cnf2bhv-ge-llength by simp**
with $\langle x < \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$ **show ?thesis**
using $\langle \neg(\exists i' \geq n'. \|c\|_t \ i') \rangle$ $\langle \|c\|_t \ i \rangle$ **le-neq-implies-less nat-le-linear by blast**

qed

ultimately obtain n''' **where** $\text{eval } c \ t \ t' \ n''' \ \gamma'$ **and** $n''' \geq \langle c \Leftarrow t \rangle_i$ **and** $n''' \leq \langle c \rightarrow t \rangle_i$
using a2 by blast

moreover from $\langle \|c\|_t \ i \rangle$ **have** $\|c\|_t \ \langle c \rightarrow t \rangle_i$ **using nxtActI by auto**

with $\langle n''' \leq \langle c \rightarrow t \rangle_i \rangle$ **have** $\exists i' \geq n'''. \|c\|_t \ i'$ **using less-or-eq-imp-le by blast**

ultimately have $\gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat} (\langle c \#_{n'''} \text{inf-llist } t \rangle))$

using validCE-act[of n''' c t t' γ'] by blast

moreover from $\langle n''' \geq \langle c \Leftarrow t \rangle_i \rangle$ **and** $\langle n''' \leq \langle c \rightarrow t \rangle_i \rangle$

have $\text{the-enat} (\langle c \#_{n'''} \text{inf-llist } t \rangle) = \text{the-enat} (\langle c \#_i \text{inf-llist } t \rangle)$ **using nAct-same by simp**

hence $\text{the-enat} (\langle c \#_{n'''} \text{inf-llist } t \rangle) = x$ **by (simp add: $\langle x = \text{the-enat} (\langle c \#_{\text{enat } i} \text{inf-llist } t \rangle) \rangle$)**

ultimately have $\gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat } x)$ **by simp**

thus ?thesis by simp

qed

qed

qed

ultimately have $\text{eval } c \ t \ t' \ n \ (\lambda \ t \ n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$

using $\langle \exists i \geq n. \|c\|_t \ i \rangle$ **validCI-act[of n c t λ t n. ∃ n'' ≥ n. γ t n'' ∧ (∀ n' ≥ n. n' < n'' → γ' t n') t']**
by blast

thus ?thesis using until-def by simp

qed

lemma untilIN[intro]:

fixes $c::'id$

and $t::\text{nat} \Rightarrow \text{cnf}$

and $t'::\text{nat} \Rightarrow 'cmp$

and $n::\text{nat}$

and $n'::\text{nat}$

assumes $\neg(\exists i \geq n. \|c\|_t \ i)$

and $n' \geq n$

and $\text{eval } c \ t \ t' \ n' \ \gamma$

and $a1: \bigwedge n''. \llbracket n \leq n''; n'' < n \rrbracket \implies \text{eval } c \ t \ t' \ n'' \ \gamma'$

shows $\text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{U}_b \ \gamma)$

proof cases

assume $\exists i. \|c\|_t \ i$

moreover from $\text{assms}(1)$ $\text{assms}(2)$ **have** $\neg(\exists i' \geq n'. \|c\|_t \ i')$ **by simp**

ultimately have $\gamma (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t (n'))$

using validCE-cont[of c t n' t' γ] $\langle \text{eval } c \ t \ t' \ n' \ \gamma \rangle$ by blast

moreover from $\langle n' \geq n \rangle$ **have** $c \downarrow_t (n') \geq c \downarrow_t (n)$ **using cnf2bhv-mono by simp**

moreover have $\forall x::\text{nat} \geq c \downarrow_t (n). x < c \downarrow_t (n') \longrightarrow \gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) x$

proof (rule HOL.all[OF HOL.impI[OF HOL.impI]])

fix x **assume** $x \geq c \downarrow_t (n)$ **and** $x < c \downarrow_t (n')$

from $\langle \neg(\exists i \geq n. \|c\|_t \ i) \rangle$ **have** $\langle c \wedge t \rangle \leq n$ **using** $\langle \exists i. \|c\|_t \ i \rangle$ **lActive-less by auto**

with $\langle x \geq c \downarrow_t(n) \rangle$ **have** $c \uparrow_t(x) \geq n$ **using** *p2c-mono-c2p* **by** *simp*
moreover from $\langle \langle c \wedge t \rangle \leq n \rangle \langle c \downarrow_t(n) \leq x \rangle$ **have** $x \geq \text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t))) - 1$
using *cnf2bhv-ge-llength dual-order.trans* **by** *blast*
with $\langle x < c \downarrow_t(n) \rangle$ **have** $c \uparrow_t(x) < n'$ **using** *c2p-mono-p2c-strict[of c t x n']* **by** *simp*
moreover from $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle \langle c \uparrow_t(x) \geq n \rangle$ **have** $\neg (\exists i'' \geq c \uparrow_t(x). \|c\|_t i'')$ **by** *auto*
ultimately have $\text{eval } c \ t \ t' \ (c \uparrow_t(x)) \ \gamma'$ **using** *a1[of c \uparrow_t(x)]* **by** *simp*
with $\langle \neg (\exists i'' \geq c \uparrow_t(x). \|c\|_t i'') \rangle$
have $\gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t(c \uparrow_t(x)))$
using *validCE-cont[of c t c \uparrow_t(x) t' \gamma']* $\langle \exists i. \|c\|_t i \rangle$ **by** *blast*
moreover have $x \geq \text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t))) - 1$
using $\langle c \downarrow_t(n) \leq x \rangle$ *cnf2bhv-def* **by** *auto*
ultimately show $\gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (x)$
using *cnf2bhv-bhv2cnf* **by** *simp*

qed

ultimately have $\text{eval } c \ t \ t' \ n \ (\lambda t \ n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$
using *validCI-cont[of c t n \lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n') \ t']*
 $\langle \exists i. \|c\|_t i \rangle \langle \neg (\exists i' \geq n. \|c\|_t i') \rangle$ **by** *blast*

thus *?thesis* **using** *until-def* **by** *simp*

next

assume $\neg (\exists i. \|c\|_t i)$

with *assms* **have** $\exists n'' \geq n. \gamma (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n'' \wedge$

$(\forall n' \geq n. n' < n'' \longrightarrow \gamma' (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n')$ **using** *validCE-not-act* **by** *blast*

with $\langle \neg (\exists i. \|c\|_t i) \rangle$ **have** $\text{eval } c \ t \ t' \ n \ (\lambda t \ n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$

using *validCI-not-act[where \gamma = \lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n')]* **by** *blast*

thus *?thesis* **using** *until-def* **by** *simp*

qed

lemma *untilEA[elim]*:

fixes $n::\text{nat}$

and $n'::\text{nat}$

and $t::\text{nat} \Rightarrow \text{cnf}$

and $t'::\text{nat} \Rightarrow \text{'cmp}$

and $c::\text{'id}$

assumes $\exists i \geq n. \|c\|_t i$

and $\text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{L}_b \ \gamma)$

shows $\exists n' \geq \langle c \rightarrow t \rangle_n.$

$(\langle \exists i \geq n'. \|c\|_t i \rangle \wedge (\forall n'' \geq \langle c \Leftarrow t \rangle_{n'}. n'' \leq \langle c \rightarrow t \rangle_{n'} \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma)$

$\wedge (\forall n'' \geq \langle c \Leftarrow t \rangle_n. n'' < \langle c \Leftarrow t \rangle_{n'} \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma') \vee$

$(\neg (\exists i \geq n'. \|c\|_t i)) \wedge \text{eval } c \ t \ t' \ n' \ \gamma \wedge (\forall n'' \geq \langle c \Leftarrow t \rangle_n. n'' < n' \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma')$

proof –

from $\langle \text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{L}_b \ \gamma) \rangle$

have $\text{eval } c \ t \ t' \ n \ (\lambda t \ n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$ **using** *until-def* **by** *simp*

with $\langle \exists i \geq n. \|c\|_t i \rangle$ **obtain** x

where $x \geq \text{the-enat} \langle c \#_{\text{enat } n} \text{inf-llist } t \rangle$ **and** $\gamma (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ x$

and $a1: \forall x' \geq \text{the-enat} \langle c \#_{\text{enat } n} \text{inf-llist } t \rangle. x' < x \longrightarrow \gamma' (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ x'$

using *validCE-act[where \gamma = \lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n')]* **by** *blast*

thus *?thesis*

proof (*cases*)

assume $x \geq \text{llength} (\pi_c(\text{inf-llist } t))$

moreover from $\langle x \geq \text{llength} (\pi_c(\text{inf-llist } t)) \rangle$ **have** $\text{llength} (\pi_c(\text{inf-llist } t)) \neq \infty$

by (*metis infinity-ileE*)

moreover from $\langle \exists i \geq n. \|c\|_t i \rangle$ **have** $\text{llength} (\pi_c(\text{inf-llist } t)) \geq 1$

using *proj-one[of inf-llist t]* **by** *auto*

ultimately have $\text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t))) - 1 < x$

by (*metis One-nat-def Suc-ile-eq antisym-conv2 diff-Suc-less enat-ord-simps(2)*)

enat-the-enat less-imp-diff-less one-enat-def
 hence $x = c \downarrow_t (c \uparrow_t(x))$ **using** *cnf2bhv-bhv2cnf* **by** *simp*
 with $\langle \gamma (l\text{nth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) x \rangle$
 have $\gamma (l\text{nth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t (c \uparrow_t(x)))$ **by** *simp*
 moreover have $\neg(\exists i \geq c \uparrow_t(x). \|c\|_t i)$
proof –
 from $\langle x \geq l\text{length } (\pi_c(\text{inf-llist } t)) \rangle$ **have** *lfinite* $(\pi_c(\text{inf-llist } t))$
 using *llength-geq-enat-lfiniteD*[of $\pi_c(\text{inf-llist } t)$ x] **by** *simp*
 then obtain z where $\forall n'' > z. \neg \|c\|_t n''$ **using** *proj-finite-bound* **by** *blast*
 moreover from $\langle \text{the-enat } (l\text{length } (\pi_c(\text{inf-llist } t))) - 1 < x \rangle$ **have** $\langle c \wedge t \rangle < c \uparrow_t(x)$
 using *bhv2cnf-greater-lActive* **by** *simp*
 ultimately show *?thesis* **using** *lActive-greater-active-all* **by** *simp*
qed
 ultimately have *eval* $c \ t \ t' (c \uparrow_t x) \ \gamma$
 using $\langle \exists i \geq n. \|c\|_t i \rangle$ *validCI-cont*[of $c \ t \ c \uparrow_t(x)$] **by** *blast*
 moreover have $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$
proof –
 from $\langle x \geq l\text{length } (\pi_c(\text{inf-llist } t)) \rangle$ **have** *lfinite* $(\pi_c(\text{inf-llist } t))$
 using *llength-geq-enat-lfiniteD*[of $\pi_c(\text{inf-llist } t)$ x] **by** *simp*
 then obtain z where $\forall n'' > z. \neg \|c\|_t n''$ **using** *proj-finite-bound* **by** *blast*
 moreover from $\langle \exists i \geq n. \|c\|_t i \rangle$ **have** $\|c\|_t \langle c \rightarrow t \rangle_n$ **using** *nextActI* **by** *simp*
 ultimately have $\langle c \wedge t \rangle \geq \langle c \rightarrow t \rangle_n$ **using** *lActive-greatest* **by** *fastforce*
 moreover have $c \uparrow_t(x) \geq \langle c \wedge t \rangle$ **by** *simp*
 ultimately show $c \uparrow_t(x) \geq \langle c \rightarrow t \rangle_n$ **by** *arith*
qed
 moreover have $\forall n'' \geq \langle c \Leftarrow t \rangle_n. n'' < (c \uparrow_t x) \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma'$
proof
 fix n'' **show** $\langle c \Leftarrow t \rangle_n \leq n'' \longrightarrow n'' < c \uparrow_t x \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma'$
proof (*rule HOL.impI[OF HOL.impI]*)
 assume $\langle c \Leftarrow t \rangle_n \leq n''$ and $n'' < c \uparrow_t x$
 show *eval* $c \ t \ t' \ n'' \ \gamma'$
proof *cases*
 assume $\exists i \geq n''. \|c\|_t i$
 with $\langle n'' \geq \langle c \Leftarrow t \rangle_n \rangle$ **have** *the-enat* $(\langle c \#_{n''} \text{inf-llist } t \rangle) \geq \text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)$
 using *nAct-mono-lNact* $\langle \exists i \geq n. \|c\|_t i \rangle$ **by** *simp*
 moreover have *the-enat* $(\langle c \#_{n''} \text{inf-llist } t \rangle) < x$
proof –
 from $\langle \exists i \geq n''. \|c\|_t i \rangle$ **have** *eSuc* $\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle \leq l\text{length } (\pi_c \text{inf-llist } t)$
 using *nAct-llength-proj* **by** *auto*
 with $\langle x \geq l\text{length } (\pi_c(\text{inf-llist } t)) \rangle$ **have** *eSuc* $\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle \leq x$ **by** *simp*
 moreover have $\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle \neq \infty$ **by** *simp*
 ultimately have *Suc* $(\text{the-enat}(\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle)) \leq x$
 by (*metis enat.distinct(2) the-enat.simps the-enat-eSuc the-enat-mono*)
 thus *?thesis* **by** *simp*
qed
 ultimately have $\gamma' (l\text{nth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat } (\langle c \#_{n''} \text{inf-llist } t \rangle))$
 using *a1* **by** *auto*
 with $\langle \exists i \geq n''. \|c\|_t i \rangle$ **show** *?thesis* **using** *validCI-act* **by** *blast*
next
 assume $\neg(\exists i \geq n''. \|c\|_t i)$
 moreover have $c \downarrow_t(n'') \geq \text{the-enat } (\langle c \#_n \text{inf-llist } t \rangle)$
proof –
 have $\langle c \#_n \text{inf-llist } t \rangle \leq l\text{length } (\pi_c(\text{inf-llist } t))$ **using** *nAct-le-proj* **by** *metis*
 moreover from $\langle \neg(\exists i \geq n''. \|c\|_t i) \rangle$ **have** $l\text{length } (\pi_c(\text{inf-llist } t)) \neq \infty$
 by (*metis llength-eq-infnty-conv-lfinite lnth-inf-llist proj-finite2*)

ultimately have $\text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle) \leq \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t)))$ by *simp*
 moreover from $\langle \exists i \geq n. \|c\|_t i \rangle \langle \neg(\exists i \geq n''. \|c\|_t i) \rangle$ have $n'' > \langle c \wedge t \rangle$
 using *lActive-active* by (*meson leI le-eq-less-or-eq*)
 hence $c \downarrow_t(n'') > \text{the-enat}(\text{llength}(\pi_c(\text{inf-llist } t))) - 1$ using *cnf2bhv-greater-llength* by *simp*
 ultimately show *?thesis* by *simp*
 qed
 moreover from $\langle \neg(\exists i \geq n''. \|c\|_t i) \rangle$ have $\langle c \wedge t \rangle \leq n''$ using *assms(1) lActive-less* by *auto*
 with $\langle n'' < c \uparrow_t x \rangle$ have $c \downarrow_t(n'') < x$ using *p2c-mono-c2p-strict* by *simp*
 ultimately have $\gamma'(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l(\text{inf-llist } t')))(c \downarrow_t(n''))$
 using *a1* by *auto*
 with $\langle \exists i \geq n. \|c\|_t i \rangle \langle \neg(\exists i \geq n''. \|c\|_t i) \rangle$ show *?thesis* using *validCI-cont* by *blast*
 qed
 qed
 qed
 ultimately show *?thesis* using $\langle \neg(\exists i \geq c \uparrow_t(x). \|c\|_t i) \rangle$ by *blast*
 next
 assume $\neg(x \geq \text{llength}(\pi_c(\text{inf-llist } t)))$
 hence $x < \text{llength}(\pi_c(\text{inf-llist } t))$ by *simp*
 then obtain $n'::\text{nat}$ where $x = \langle c \#_{n'} \text{inf-llist } t \rangle$ using *nAct-exists* by *blast*
 with $\langle \text{enat } x < \text{llength}(\pi_c(\text{inf-llist } t)) \rangle$ have $\exists i \geq n'. \|c\|_t i$ using *nAct-less-llength-active* by *force*
 then obtain i where $i \geq n'$ and $\|c\|_t i$ and $\neg(\exists k \geq n'. k < i \wedge \|c\|_t k)$ using *nact-exists* by *blast*
 moreover have $(\forall n'' \geq \langle c \leftarrow t \rangle_i. n'' \leq \langle c \rightarrow t \rangle_i \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma)$
 proof
 fix n'' show $\langle c \leftarrow t \rangle_i \leq n'' \longrightarrow n'' \leq \langle c \rightarrow t \rangle_i \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma$
 proof(*rule HOL.impI[OF HOL.impI]*)
 assume $\langle c \leftarrow t \rangle_i \leq n''$ and $n'' \leq \langle c \rightarrow t \rangle_i$
 hence $\text{the-enat}(\langle c \#_{\text{enat } i} \text{inf-llist } t \rangle) = \text{the-enat}(\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle)$
 using *nAct-same* by *simp*
 moreover from $\langle \|c\|_t i \rangle$ have $\|c\|_t \langle c \rightarrow t \rangle_i$ using *nextActI* by *auto*
 with $\langle n'' \leq \langle c \rightarrow t \rangle_i \rangle$ have $\exists i \geq n''. \|c\|_t i$ using *dual-order.strict-implies-order* by *auto*
 moreover have $\gamma(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l(\text{inf-llist } t')))(\text{the-enat}(\langle c \#_{\text{enat } i} \text{inf-llist } t \rangle))$
 proof –
 have $\text{enat } i - 1 < \text{llength}(\text{inf-llist } t)$ by (*simp add: one-enat-def*)
 with $\langle x = \langle c \#_{n'} \text{inf-llist } t \rangle \langle i \geq n' \rangle \langle \neg(\exists k \geq n'. k < i \wedge \|c\|_t k) \rangle$ have $x = \langle c \#_i \text{inf-llist } t \rangle$
 using *one-enat-def nAct-not-active-same* by *simp*
 moreover have $\langle c \#_i \text{inf-llist } t \rangle \neq \infty$ by *simp*
 ultimately have $x = \text{the-enat}(\langle c \#_i \text{inf-llist } t \rangle)$ by *fastforce*
 thus *?thesis* using $\langle \gamma(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l(\text{inf-llist } t')) \ x) \rangle$ by *blast*
 qed
 with $\langle \text{the-enat}(\langle c \#_{\text{enat } i} \text{inf-llist } t \rangle) = \text{the-enat}(\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle) \rangle$ have
 $\gamma(\text{lnth}((\pi_c(\text{inf-llist } t)) @_l(\text{inf-llist } t')))(\text{the-enat}(\langle c \#_{\text{enat } n''} \text{inf-llist } t \rangle))$ by *simp*
 ultimately show $\text{eval } c \ t \ t' \ n'' \ \gamma$ using *validCI-act* by *blast*
 qed
 qed
 moreover have $i \geq \langle c \rightarrow t \rangle_n$
 proof –
 have $\text{enat } i - 1 < \text{llength}(\text{inf-llist } t)$ by (*simp add: one-enat-def*)
 with $\langle x = \langle c \#_{n'} \text{inf-llist } t \rangle \langle i \geq n' \rangle \langle \neg(\exists k \geq n'. k < i \wedge \|c\|_t k) \rangle$ have $x = \langle c \#_i \text{inf-llist } t \rangle$
 using *one-enat-def nAct-not-active-same* by *simp*
 moreover have $\langle c \#_i \text{inf-llist } t \rangle \neq \infty$ by *simp*
 ultimately have $x = \text{the-enat}(\langle c \#_i \text{inf-llist } t \rangle)$ by *fastforce*
 with $\langle x \geq \text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle) \rangle$
 have $\text{the-enat}(\langle c \#_i \text{inf-llist } t \rangle) \geq \text{the-enat}(\langle c \#_n \text{inf-llist } t \rangle)$ by *simp*
 with $\langle \|c\|_t i \rangle$ show *?thesis* using *active-geq-nextAct* by *simp*
 qed

moreover have $\forall n'' \geq \langle c \Leftarrow t \rangle_n. n'' < \langle c \Leftarrow t \rangle_i \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma'$

proof

fix n'' **show** $\langle c \Leftarrow t \rangle_n \leq n'' \longrightarrow n'' < \langle c \Leftarrow t \rangle_i \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma'$

proof (rule *HOL.impI*[*OF HOL.impI*])

assume $\langle c \Leftarrow t \rangle_n \leq n''$ **and** $n'' < \langle c \Leftarrow t \rangle_i$

moreover have $\langle c \Leftarrow t \rangle_{i \leq i}$ **by** *simp*

ultimately have $\exists i \geq n''.$ $\|c\|_{t \ i}$ **using** $\langle \|c\|_{t \ i} \rangle$ **by** (*meson less-le less-le-trans*)

with $\langle n'' \geq \langle c \Leftarrow t \rangle_n \rangle$ **have** *the-enat* ($\langle c \#_{n''} \text{inf-llist } t \rangle$) \geq *the-enat* ($\langle c \#_n \text{inf-llist } t \rangle$)

using *nAct-mono-lNact* $\langle \exists i \geq n. \|c\|_{t \ i} \rangle$ **by** *simp*

moreover have *the-enat* ($\langle c \#_{n''} \text{inf-llist } t \rangle$) $< x$

proof –

from $\langle n'' < \langle c \Leftarrow t \rangle_i \rangle$ $\langle \langle c \Leftarrow t \rangle_i \leq i \rangle$ **have** $n'' < i$ **using** *dual-order.strict-trans1* **by** *arith*

with $\langle n'' < \langle c \Leftarrow t \rangle_i \rangle$ **have** $\exists i' \geq n''.$ $i' < i \wedge \|c\|_{t \ i'}$ **using** *lNact-least*[*of i n''*] **by** *fastforce*

hence $\langle c \#_{n''} \text{inf-llist } t \rangle < \langle c \#_i \text{inf-llist } t \rangle$ **using** *nAct-less* **by** *auto*

moreover have *enat* $i - 1 < \text{llength } (\text{inf-llist } t)$ **by** (*simp add: one-enat-def*)

with $\langle x = \langle c \#_{n'} \text{inf-llist } t \rangle \rangle$ $\langle i \geq n' \rangle$ $\langle \neg (\exists k \geq n'. k < i \wedge \|c\|_{t \ k}) \rangle$ **have** $x = \langle c \#_i \text{inf-llist } t \rangle$

using *one-enat-def nAct-not-active-same* **by** *simp*

moreover have $\langle c \#_{n''} \text{inf-llist } t \rangle \neq \infty$ **by** *simp*

ultimately show *?thesis* **by** (*metis enat-ord-simps(2) enat-the-enat*)

qed

ultimately have $\gamma' (\text{lth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (\text{the-enat } (\langle c \#_{n''} \text{inf-llist } t \rangle))$

using *a1* **by** *auto*

with $\langle \exists i \geq n''.$ $\|c\|_{t \ i} \rangle$ **show** $\text{eval } c \ t \ t' \ n'' \ \gamma'$ **using** *validCI-act* **by** *blast*

qed

qed

ultimately show *?thesis* **using** $\langle \|c\|_{t \ i} \rangle$ **by** *auto*

qed

qed

lemma *untilEN*[*elim*]:

fixes $n::\text{nat}$

and $n'::\text{nat}$

and $t::\text{nat} \Rightarrow \text{cnf}$

and $t'::\text{nat} \Rightarrow \text{'cmp}$

and $c::\text{'id}$

assumes $\#i. i \geq n \wedge \|c\|_{t \ i}$

and $\text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{L}_b \ \gamma)$

shows $\exists n' \geq n. \text{eval } c \ t \ t' \ n' \ \gamma \wedge$

$(\forall n'' \geq n. n'' < n' \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma')$

proof *cases*

assume $\exists i. \|c\|_{t \ i}$

moreover from $\langle \text{eval } c \ t \ t' \ n \ (\gamma' \ \mathfrak{L}_b \ \gamma) \rangle$

have $\text{eval } c \ t \ t' \ n \ (\lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$ **using** *until-def* **by** *simp*

ultimately have $\exists n'' \geq c \downarrow_t(n). \gamma (\text{lth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n'' \wedge$

$(\forall n' \geq c \downarrow_t(n). n' < n'' \longrightarrow \gamma' (\text{lth } (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n')$

using *validCE-cont*[**where** $\gamma = \lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n')$]

$\langle \#i. i \geq n \wedge \|c\|_{t \ i} \rangle$ **by** *blast*

then obtain x **where** $x \geq c \downarrow_t(n)$ **and** $\gamma (\text{lth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) \ x$

and $\forall x' \geq c \downarrow_t(n). x' < x \longrightarrow \gamma' (\text{lth } ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) \ x'$ **by** *auto*

moreover from $\langle \neg (\exists i \geq n. \|c\|_{t \ i}) \rangle$ **have** *the-enat* ($\text{llength } (\pi_c(\text{inf-llist } t))$) $- 1 < x$

proof –

have $\langle c \wedge t \rangle < n$

proof (rule *ccontr*)

assume $\neg \langle c \wedge t \rangle < n$

hence $\langle c \wedge t \rangle \geq n$ **by** *simp*

moreover from $\langle \exists i. \|c\|_t i \rangle \langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ **have** $\|c\|_t \langle c \wedge t \rangle$
using *LActive-active less-or-eq-imp-le* **by** *blast*
ultimately show *False* **using** $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ **by** *simp*
qed
hence *the-enat* ($\text{llength} (\pi_c(\text{inf-llist } t)) - 1 < c \downarrow_t(n)$) **using** *cnf2bhv-greater-llength* **by** *simp*
with $\langle x \geq c \downarrow_t(n) \rangle$ **show** *?thesis* **by** *simp*
qed
hence $x = c \downarrow_t(c \uparrow_t(x))$ **using** *cnf2bhv-bhv2cnf* **by** *simp*
ultimately have $\gamma (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t(c \uparrow_t(x)))$ **by** *simp*
moreover from $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ **have** $\neg (\exists i \geq c \uparrow_t(x). \|c\|_t i)$
proof –
from $\langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ **have** *lfinite* ($\pi_c(\text{inf-llist } t)$) **using** *proj-finite2* **by** *simp*
then obtain z **where** $\forall n'' > z. \neg \|c\|_t n''$ **using** *proj-finite-bound* **by** *blast*
moreover from $\langle \text{the-enat} (\text{llength} (\pi_c(\text{inf-llist } t)) - 1 < x) \rangle$ **have** $\langle c \wedge t \rangle < c \uparrow_t(x)$
using *bhv2cnf-greater-LActive* **by** *simp*
ultimately show *?thesis* **using** *LActive-greater-active-all* **by** *simp*
qed
ultimately have *eval* $c \ t \ t' (c \uparrow_t(x)) \ \gamma$ **using** *validCI-cont* $\langle \exists i. \|c\|_t i \rangle$ **by** *blast*
moreover from $\langle \exists i. \|c\|_t i \rangle \langle \neg (\exists i \geq n. \|c\|_t i) \rangle$ **have** $\langle c \wedge t \rangle \leq n$ **using** *LActive-less[of c t - n]* **by** *auto*
with $\langle x \geq c \downarrow_t(n) \rangle$ **have** $n \leq c \uparrow_t(x)$ **using** *p2c-mono-c2p* **by** *blast*
moreover have $\forall n'' \geq n. n'' < c \uparrow_t(x) \longrightarrow \text{eval } c \ t \ t' \ n'' \ \gamma'$
proof (*rule HOL.all[OF HOL.impI[OF HOL.impI]*)
fix n'' **assume** $n \leq n''$ **and** $n'' < c \uparrow_t(x)$
hence $c \downarrow_t(n'') \geq c \downarrow_t(n)$ **using** *cnf2bhv-mono* **by** *simp*
moreover have $n'' < c \uparrow_t(x)$ **by** (*simp add: $\langle n'' < c \uparrow_t(x) \rangle$*)
with $\langle \langle c \wedge t \rangle \leq n \rangle \langle n \leq n'' \rangle$ **have** $c \downarrow_t(n'') < c \downarrow_t(c \uparrow_t(x))$ **using** *cnf2bhv-mono-strict* **by** *simp*
with $\langle x = c \downarrow_t(c \uparrow_t(x)) \rangle$ **have** $c \downarrow_t(n'') < x$ **by** *simp*
ultimately have $\gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) (c \downarrow_t(n''))$
using $\langle \forall x' \geq c \downarrow_t(n). x' < x \longrightarrow \gamma' (\text{lnth} ((\pi_c(\text{inf-llist } t)) @_l (\text{inf-llist } t'))) x' \rangle$ **by** *simp*
moreover from $\langle n \leq n'' \rangle$ **have** $\nexists i. i \geq n'' \wedge \|c\|_t i$ **using** $\langle \nexists i. i \geq n \wedge \|c\|_t i \rangle$ **by** *simp*
ultimately show *eval* $c \ t \ t' \ n'' \ \gamma'$ **using** *validCI-cont* **using** $\langle \exists i. \|c\|_t i \rangle$ **by** *blast*
qed
ultimately show *?thesis* **by** *auto*
next
assume $\neg (\exists i. \|c\|_t i)$
moreover from $\langle \text{eval } c \ t \ t' \ n (\gamma' \ \mathfrak{L}_b \ \gamma) \rangle$
have *eval* $c \ t \ t' \ n (\lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n'))$ **using** *until-def* **by** *simp*
ultimately have $\exists n'' \geq n. \gamma (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n''$
 $\wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' (\text{lnth} (\pi_c \text{inf-llist } t @_l \text{inf-llist } t')) \ n')$ **using** $\langle \neg (\exists i. \|c\|_t i) \rangle$
validCE-not-act [**where** $\gamma = \lambda t n. \exists n'' \geq n. \gamma \ t \ n'' \wedge (\forall n' \geq n. n' < n'' \longrightarrow \gamma' \ t \ n')$] **by** *blast*
with $\langle \neg (\exists i. \|c\|_t i) \rangle$ **show** *?thesis* **using** *validCI-not-act* **by** *blast*
qed

2.4.13 Weak Until

definition *wuntil* $:: ('cmp \ bta) \Rightarrow ('cmp \ bta) \Rightarrow ('cmp \ bta) \ (\text{infixl } \mathfrak{W}_b \ 20)$
where $\gamma' \ \mathfrak{W}_b \ \gamma \equiv \gamma' \ \mathfrak{L}_b \ \gamma \ \vee^b \ \square_b(\gamma')$

end

end

References

- [1] A. Lochbihler. Coinduction. *The Archive of Formal Proofs*. <https://isa-afp.org/entries/Coinductive.shtml>, 2010.
- [2] D. Marmsoler. On the semantics of temporal specifications of component-behavior for dynamic architectures. In *Eleventh International Symposium on Theoretical Aspects of Software Engineering*. Springer, 2017.
- [3] D. Marmsoler. Towards a calculus for dynamic architectures. In *International Colloquium on Theoretical Aspects of Computing*. Springer, 2017.
- [4] D. Marmsoler and M. Gleirscher. On activation, connection, and behavior in dynamic architectures. *Scientific Annals of Computer Science*, 26(2):187–248, 2016.
- [5] D. Marmsoler and M. Gleirscher. Specifying properties of dynamic architectures using configuration traces. In *International Colloquium on Theoretical Aspects of Computing*, pages 235–254. Springer, 2016.