

# Analysing and Comparing Encodability Criteria for Process Calculi (Technical Report)

Kirstin Peters\*  
TU Dresden, Germany

Rob van Glabbeek  
NICTA<sup>†</sup> Sydney, Australia  
Computer Science and Engineering, UNSW, Sydney, Australia

August 05, 2015

## Abstract

Encodings or the proof of their absence are the main way to compare process calculi. To analyse the quality of encodings and to rule out trivial or meaningless encodings, they are augmented with quality criteria. There exists a bunch of different criteria and different variants of criteria in order to reason in different settings. This leads to incomparable results. Moreover it is not always clear whether the criteria used to obtain a result in a particular setting do indeed fit to this setting. We show how to formally reason about and compare encodability criteria by mapping them on requirements on a relation between source and target terms that is induced by the encoding function. In particular we analyse the common criteria *full abstraction*, *operational correspondence*, *divergence reflection*, *success sensitiveness*, and *respect of barbs*; e.g. we analyse the exact nature of the simulation relation (coupled simulation versus bisimulation) that is induced by different variants of operational correspondence. This way we reduce the problem of analysing or comparing encodability criteria to the better understood problem of comparing relations on processes.

In the following we present the Isabelle implementation of the underlying theory as well as all proofs of the results presented in the paper *Analysing and Comparing Encodability Criteria* as submitted to EXPRESS/SOS'15.

---

\*Supported by funding of the Excellence Initiative by the German Federal and State Governments (Institutional Strategy, measure 'support the best').

<sup>†</sup>NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.

# Contents

<b>1</b>	<b>Relations</b>	<b>3</b>
1.1	Basic Conditions . . . . .	3
1.2	Preservation, Reflection, and Respection of Predicates . . . . .	6
<b>2</b>	<b>Process Calculi</b>	<b>11</b>
2.1	Reduction Semantics . . . . .	11
2.1.1	Observables or Barbs . . . . .	13
<b>3</b>	<b>Simulation Relations</b>	<b>17</b>
3.1	Simulation . . . . .	17
3.2	Contrasimulation . . . . .	22
3.3	Coupled Simulation . . . . .	24
3.4	Correspondence Simulation . . . . .	25
3.5	Bisimulation . . . . .	30
3.6	Step Closure of Relations . . . . .	40
<b>4</b>	<b>Encodings</b>	<b>46</b>
<b>5</b>	<b>Relation between Source and Target Terms</b>	<b>54</b>
5.1	Relations Induced by the Encoding Function . . . . .	54
5.2	Relations Induced by the Encoding and a Relation on Target Terms . . . . .	77
5.3	Relations Induced by the Encoding and Relations on Source Terms and Target Terms . . . . .	120
<b>6</b>	<b>Success Sensitiveness and Barbs</b>	<b>151</b>
<b>7</b>	<b>Divergence Reflection</b>	<b>155</b>
<b>8</b>	<b>Operational Correspondence</b>	<b>157</b>
8.1	Trivial Operational Correspondence Results . . . . .	158
8.2	(Strong) Operational Completeness vs (Strong) Simulation . . . . .	159
8.3	Weak Operational Soundness vs Contrasimulation . . . . .	167
8.4	(Strong) Operational Soundness vs (Strong) Simulation . . . . .	168
8.5	Weak Operational Correspondence vs Correspondence Similarity . . . . .	175
8.6	(Strong) Operational Correspondence vs (Strong) Bisimilarity . . . . .	181
<b>9</b>	<b>Full Abstraction</b>	<b>201</b>
9.1	Trivial Full Abstraction Results . . . . .	201
9.2	Fully Abstract Encodings . . . . .	203
9.3	Full Abstraction w.r.t. Preorders . . . . .	210
9.4	Full Abstraction w.r.t. Equivalences . . . . .	218
9.5	Full Abstraction without Relating Translations to their Source Terms . . . . .	228
<b>10</b>	<b>Combining Criteria</b>	<b>235</b>
10.1	Divergence Reflection and Success Sensitiveness . . . . .	237
10.2	Adding Operational Correspondence . . . . .	238
10.3	Full Abstraction and Operational Correspondence . . . . .	271

```

theory Relations
  imports Main HOL-Library.LaTeXsugar HOL-Library.OptionalSugar
begin

```

# 1 Relations

## 1.1 Basic Conditions

We recall the standard definitions for reflexivity, symmetry, transitivity, preorders, equivalence, and inverse relations.

**abbreviation** *preorder Rel*  $\equiv$  *preorder-on UNIV Rel*

**abbreviation** *equivalence Rel*  $\equiv$  *equiv UNIV Rel*

A symmetric preorder is an equivalence.

**lemma** *symm-preorder-is-equivalence*:

**fixes** *Rel* :: ('a × 'a) set

**assumes** *preorder Rel*

**and** *sym Rel*

**shows** *equivalence Rel*

**using** *assms*

**unfolding** *preorder-on-def equiv-def*

**by** *simp*

The symmetric closure of a relation is the union of this relation and its inverse.

**definition** *symcl* :: ('a × 'a) set  $\Rightarrow$  ('a × 'a) set **where**

*symcl Rel* = *Rel*  $\cup$  *Rel*<sup>-1</sup>

For all (a, b) in R, the symmetric closure of R contains (a, b) as well as (b, a).

**lemma** *elem-of-symcl*:

**fixes** *Rel* :: ('a × 'a) set

**and** *a b* :: 'a

**assumes** *elem*: (a, b)  $\in$  *Rel*

**shows** (a, b)  $\in$  *symcl Rel*

**and** (b, a)  $\in$  *symcl Rel*

**by** (*auto simp add: elem symcl-def*)

The symmetric closure of a relation is symmetric.

**lemma** *sym-symcl*:

**fixes** *Rel* :: ('a × 'a) set

**shows** *sym (symcl Rel)*

**by** (*simp add: symcl-def sym-Un-converse*)

The reflexive and symmetric closure of a relation is equal to its symmetric and reflexive closure.

**lemma** *refl-symm-closure-is-symm-refl-closure*:

**fixes** *Rel* :: ('a × 'a) set

**shows** *symcl (Rel<sup>=</sup>)* = (*symcl Rel*)<sup>=</sup>

**by** (*auto simp add: symcl-def refl*)

The symmetric closure of a reflexive relation is reflexive.

**lemma** *refl-symcl-of-refl-rel*:

**fixes** *Rel* :: ('a × 'a) set

**and** *A* :: 'a set

**assumes** *refl-on A Rel*

**shows** *refl-on A (symcl Rel)*

**using** *assms*

**by** (*auto simp add: refl-on-def' symcl-def*)

Accordingly, the reflexive, symmetric, and transitive closure of a relation is equal to its symmetric, reflexive, and transitive closure.

**lemma** *refl-symm-trans-closure-is-symm-refl-trans-closure*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**shows**  $(\text{symcl } (Rel^=))^+ = (\text{symcl } Rel)^*$   
**using** *refl-symm-closure-is-symm-refl-closure*[**where**  $Rel=Rel$ ]  
**by** *simp*

The reflexive closure of a symmetric relation is symmetric.

**lemma** *sym-reflcl-of-symm-rel*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**assumes** *sym Rel*  
**shows**  $\text{sym } (Rel^=)$   
**using** *assms*  
**by** (*simp add: sym-Id sym-Un*)

The reflexive closure of a reflexive relation is the relation itself.

**lemma** *reflcl-of-refl-rel*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**assumes** *refl Rel*  
**shows**  $Rel^= = Rel$   
**using** *assms*  
**unfolding** *refl-on-def*  
**by** *auto*

The symmetric closure of a symmetric relation is the relation itself.

**lemma** *symm-closure-of-symm-rel*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**assumes** *sym Rel*  
**shows**  $\text{symcl } Rel = Rel$   
**using** *assms*  
**unfolding** *symcl-def sym-def*  
**by** *auto*

The reflexive and transitive closure of a preorder  $Rel$  is  $Rel$ .

**lemma** *rtrancl-of-preorder*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**assumes** *preorder Rel*  
**shows**  $Rel^* = Rel$   
**using** *assms reflcl-of-refl-rel*[*of Rel*] *trancl-id*[*of Rel^=*] *trancl-reflcl*[*of Rel*]  
**unfolding** *preorder-on-def*  
**by** *auto*

The reflexive and transitive closure of a relation is a subset of its reflexive, symmetric, and transitive closure.

**lemma** *refl-trans-closure-subset-of-refl-symm-trans-closure*:  
**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**shows**  $Rel^* \subseteq (\text{symcl } (Rel^=))^+$   
**proof** *clarify*  
**fix**  $a b$   
**assume**  $(a, b) \in Rel^*$   
**hence**  $(a, b) \in (\text{symcl } Rel)^*$   
**using** *in-rtrancl-UnI*[*of (a, b) Rel Rel<sup>-1</sup>*]  
**by** (*simp add: symcl-def*)  
**thus**  $(a, b) \in (\text{symcl } (Rel^=))^+$   
**using** *refl-symm-trans-closure-is-symm-refl-trans-closure*[*of Rel*]  
**by** *simp*  
**qed**

If a preorder  $Rel$  satisfies the following two conditions, then its symmetric closure is transitive: (1) If  $(a, b)$  and  $(c, b)$  in  $Rel$  but not  $(a, c)$  in  $Rel$ , then  $(b, a)$  in  $Rel$  or  $(b, c)$  in  $Rel$ . (2) If  $(a, b)$  and  $(a, c)$  in  $Rel$  but not  $(b, c)$  in  $Rel$ , then  $(b, a)$  in  $Rel$  or  $(c, a)$  in  $Rel$ .

**lemma** *symm-closure-of-preorder-is-trans*:

**fixes**  $Rel :: ('a \times 'a) \text{ set}$

**assumes**  $condA: \forall a b c. (a, b) \in Rel \wedge (c, b) \in Rel \wedge (a, c) \notin Rel$   
 $\longrightarrow (b, a) \in Rel \vee (b, c) \in Rel$

**and**  $condB: \forall a b c. (a, b) \in Rel \wedge (a, c) \in Rel \wedge (b, c) \notin Rel$   
 $\longrightarrow (b, a) \in Rel \vee (c, a) \in Rel$

**and**  $reflR: refl\ Rel$

**and**  $tranR: trans\ Rel$

**shows**  $trans\ (symcl\ Rel)$

**unfolding** *trans-def*

**proof** *clarify*

**fix**  $a\ b\ c$

**have**  $\llbracket (a, b) \in Rel; (b, c) \in Rel \rrbracket \Longrightarrow (a, c) \in symcl\ Rel$

**proof**  $-$

**assume**  $(a, b) \in Rel$  **and**  $(b, c) \in Rel$

**with**  $tranR$  **have**  $(a, c) \in Rel$

**unfolding** *trans-def*

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**qed**

**moreover**  $\llbracket (a, b) \in Rel; (c, b) \in Rel; (a, c) \notin Rel \rrbracket \Longrightarrow (a, c) \in symcl\ Rel$

**proof**  $-$

**assume**  $A1: (a, b) \in Rel$  **and**  $A2: (c, b) \in Rel$  **and**  $(a, c) \notin Rel$

**with**  $condA$  **have**  $(b, a) \in Rel \vee (b, c) \in Rel$

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**proof** *auto*

**assume**  $(b, a) \in Rel$

**with**  $A2\ tranR$  **have**  $(c, a) \in Rel$

**unfolding** *trans-def*

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**next**

**assume**  $(b, c) \in Rel$

**with**  $A1\ tranR$  **have**  $(a, c) \in Rel$

**unfolding** *trans-def*

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**qed**

**moreover**  $\llbracket (b, a) \in Rel; (b, c) \in Rel; (a, c) \notin Rel \rrbracket \Longrightarrow (a, c) \in symcl\ Rel$

**proof**  $-$

**assume**  $B1: (b, a) \in Rel$  **and**  $B2: (b, c) \in Rel$  **and**  $(a, c) \notin Rel$

**with**  $condB$  **have**  $(a, b) \in Rel \vee (c, b) \in Rel$

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**proof** *auto*

**assume**  $(a, b) \in Rel$

**with**  $B2\ tranR$  **have**  $(a, c) \in Rel$

**unfolding** *trans-def*

**by** *blast*

**thus**  $(a, c) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**next**

**assume**  $(c, b) \in Rel$

**with**  $B1\ tranR$  **have**  $(c, a) \in Rel$

**unfolding** *trans-def*

**by** *blast*

```

    thus (a, c) ∈ symcl Rel
      by (simp add: symcl-def)
  qed
  moreover have  $\llbracket (b, a) \in Rel; (c, b) \in Rel \rrbracket \implies (a, c) \in symcl Rel$ 
  proof -
    assume (c, b) ∈ Rel and (b, a) ∈ Rel
    with tranR have (c, a) ∈ Rel
      unfolding trans-def
      by blast
    thus (a, c) ∈ symcl Rel
      by (simp add: symcl-def)
  qed
  moreover assume (a, b) ∈ symcl Rel and (b, c) ∈ symcl Rel
  ultimately show (a, c) ∈ symcl Rel
    by (auto simp add: symcl-def)
  qed

```

## 1.2 Preservation, Reflection, and Respection of Predicates

A relation  $R$  preserves some predicate  $P$  if  $P(a)$  implies  $P(b)$  for all  $(a, b)$  in  $R$ .

**abbreviation**  $rel\text{-preserves-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-preserves-pred } Rel \text{ Pred} \equiv \forall a \ b. (a, b) \in Rel \wedge \text{Pred } a \longrightarrow \text{Pred } b$

**abbreviation**  $rel\text{-preserves-binary-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-preserves-binary-pred } Rel \text{ Pred} \equiv \forall a \ b \ x. (a, b) \in Rel \wedge \text{Pred } a \ x \longrightarrow \text{Pred } b \ x$

A relation  $R$  reflects some predicate  $P$  if  $P(b)$  implies  $P(a)$  for all  $(a, b)$  in  $R$ .

**abbreviation**  $rel\text{-reflects-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-reflects-pred } Rel \text{ Pred} \equiv \forall a \ b. (a, b) \in Rel \wedge \text{Pred } b \longrightarrow \text{Pred } a$

**abbreviation**  $rel\text{-reflects-binary-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-reflects-binary-pred } Rel \text{ Pred} \equiv \forall a \ b \ x. (a, b) \in Rel \wedge \text{Pred } b \ x \longrightarrow \text{Pred } a \ x$

A relation respects a predicate if it preserves and reflects it.

**abbreviation**  $rel\text{-respects-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-respects-pred } Rel \text{ Pred} \equiv rel\text{-preserves-pred } Rel \text{ Pred} \wedge rel\text{-reflects-pred } Rel \text{ Pred}$

**abbreviation**  $rel\text{-respects-binary-pred} :: ('a \times 'a) \text{ set} \Rightarrow ('a \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool}$  **where**  
 $rel\text{-respects-binary-pred } Rel \text{ Pred} \equiv rel\text{-preserves-binary-pred } Rel \text{ Pred} \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred}$

For symmetric relations preservation, reflection, and respection of predicates means the same.

**lemma**  $symm\text{-relation-impl-preservation-equals-reflection}$ :

```

  fixes Rel :: ('a × 'a) set
    and Pred :: 'a ⇒ bool
  assumes symm: sym Rel
  shows rel-preserves-pred Rel Pred = rel-reflects-pred Rel Pred
    and rel-preserves-pred Rel Pred = rel-respects-pred Rel Pred
    and rel-reflects-pred Rel Pred = rel-respects-pred Rel Pred
    using symm
    unfolding sym-def
    by blast+

```

**lemma**  $symm\text{-relation-impl-preservation-equals-reflection-of-binary-predicates}$ :

```

  fixes Rel :: ('a × 'a) set
    and Pred :: 'a ⇒ 'b ⇒ bool
  assumes symm: sym Rel
  shows rel-preserves-binary-pred Rel Pred = rel-reflects-binary-pred Rel Pred

```

```

and rel-preserves-binary-pred Rel Pred = rel-respects-binary-pred Rel Pred
and rel-reflects-binary-pred Rel Pred = rel-respects-binary-pred Rel Pred
  using symm
  unfolding sym-def
by blast+

```

If a relation preserves a predicate then so does its reflexive or/and transitive closure.

**lemma** *preservation-and-closures*:

```

fixes Rel :: ('a × 'a) set
  and Pred :: 'a ⇒ bool
assumes preservation: rel-preserves-pred Rel Pred
shows rel-preserves-pred (Rel=) Pred
  and rel-preserves-pred (Rel+) Pred
  and rel-preserves-pred (Rel*) Pred
proof –
from preservation show A: rel-preserves-pred (Rel=) Pred
  by (auto simp add: refl)
have B: ∧Rel. rel-preserves-pred Rel Pred ⇒ rel-preserves-pred (Rel+) Pred
proof clarify
  fix Rel a b
  assume (a, b) ∈ Rel+ and rel-preserves-pred Rel Pred and Pred a
  thus Pred b
  by (induct, blast+)
qed
with preservation show rel-preserves-pred (Rel+) Pred
  by blast
from preservation A B[where Rel=Rel=] show rel-preserves-pred (Rel*) Pred
  using trancl-reflcl[of Rel]
  by blast
qed

```

**lemma** *preservation-of-binary-predicates-and-closures*:

```

fixes Rel :: ('a × 'a) set
  and Pred :: 'a ⇒ 'b ⇒ bool
assumes preservation: rel-preserves-binary-pred Rel Pred
shows rel-preserves-binary-pred (Rel=) Pred
  and rel-preserves-binary-pred (Rel+) Pred
  and rel-preserves-binary-pred (Rel*) Pred
proof –
from preservation show A: rel-preserves-binary-pred (Rel=) Pred
  by (auto simp add: refl)
have B: ∧Rel. rel-preserves-binary-pred Rel Pred
  ⇒ rel-preserves-binary-pred (Rel+) Pred
proof clarify
  fix Rel a b x
  assume (a, b) ∈ Rel+ and rel-preserves-binary-pred Rel Pred and Pred a x
  thus Pred b x
  by (induct, blast+)
qed
with preservation show rel-preserves-binary-pred (Rel+) Pred
  by blast
from preservation A B[where Rel=Rel=]
show rel-preserves-binary-pred (Rel*) Pred
  using trancl-reflcl[of Rel]
  by fast
qed

```

If a relation reflects a predicate then so does its reflexive or/and transitive closure.

**lemma** *reflection-and-closures*:

```

fixes Rel :: ('a × 'a) set

```

```

    and Pred :: 'a ⇒ bool
  assumes reflection: rel-reflects-pred Rel Pred
  shows rel-reflects-pred (Rel=) Pred
    and rel-reflects-pred (Rel+) Pred
    and rel-reflects-pred (Rel*) Pred
proof -
  from reflection show A: rel-reflects-pred (Rel=) Pred
    by (auto simp add: refl)
  have B:  $\bigwedge$ Rel. rel-reflects-pred Rel Pred  $\implies$  rel-reflects-pred (Rel+) Pred
  proof clarify
    fix Rel a b
    assume (a, b) ∈ Rel+ and rel-reflects-pred Rel Pred and Pred b
    thus Pred a
      by (induct, blast+)
  qed
  with reflection show rel-reflects-pred (Rel+) Pred
    by blast
  from reflection A B[where Rel=Rel=] show rel-reflects-pred (Rel*) Pred
    using trancl-reflcl[of Rel]
    by fast
qed

```

```

lemma reflection-of-binary-predicates-and-closures:
  fixes Rel :: ('a × 'a) set
    and Pred :: 'a ⇒ 'b ⇒ bool
  assumes reflection: rel-reflects-binary-pred Rel Pred
  shows rel-reflects-binary-pred (Rel=) Pred
    and rel-reflects-binary-pred (Rel+) Pred
    and rel-reflects-binary-pred (Rel*) Pred
proof -
  from reflection show A: rel-reflects-binary-pred (Rel=) Pred
    by (auto simp add: refl)
  have B:  $\bigwedge$ Rel. rel-reflects-binary-pred Rel Pred  $\implies$  rel-reflects-binary-pred (Rel+) Pred
  proof clarify
    fix Rel a b x
    assume (a, b) ∈ Rel+ and rel-reflects-binary-pred Rel Pred and Pred b x
    thus Pred a x
      by (induct, blast+)
  qed
  with reflection show rel-reflects-binary-pred (Rel+) Pred
    by blast
  from reflection A B[where Rel=Rel=]
  show rel-reflects-binary-pred (Rel*) Pred
    using trancl-reflcl[of Rel]
    by fast
qed

```

If a relation respects a predicate then so does its reflexive, symmetric, or/and transitive closure.

```

lemma respecton-and-closures:
  fixes Rel :: ('a × 'a) set
    and Pred :: 'a ⇒ bool
  assumes respecton: rel-respects-pred Rel Pred
  shows rel-respects-pred (Rel=) Pred
    and rel-respects-pred (symcl Rel) Pred
    and rel-respects-pred (Rel+) Pred
    and rel-respects-pred (symcl (Rel=)) Pred
    and rel-respects-pred (Rel*) Pred
    and rel-respects-pred ((symcl (Rel=))+) Pred
proof -
  from respecton show A: rel-respects-pred (Rel=) Pred
    using preservation-and-closures(1)[where Rel=Rel and Pred=Pred]

```



```

    reflection-and-closures(1)[where Rel=Rel and Pred=Pred]
  by blast
have B:  $\bigwedge Rel. \text{rel-respects-pred } Rel \text{ Pred} \implies \text{rel-respects-pred } (\text{symcl } Rel) \text{ Pred}$ 
proof
  fix Rel
  assume B1: rel-respects-pred Rel Pred
  show rel-preserves-pred (symcl Rel) Pred
  proof clarify
    fix a b
    assume (a, b)  $\in$  symcl Rel
    hence (a, b)  $\in$  Rel  $\vee$  (b, a)  $\in$  Rel
      by (simp add: symcl-def)
    moreover assume Pred a
    ultimately show Pred b
      using B1
    by blast
  qed
next
  fix Rel :: ('a  $\times$  'a) set
  and Pred :: 'a  $\Rightarrow$  bool
  assume B2: rel-respects-pred Rel Pred
  show rel-reflects-pred (symcl Rel) Pred
  proof clarify
    fix a b
    assume (a, b)  $\in$  symcl Rel
    hence (a, b)  $\in$  Rel  $\vee$  (b, a)  $\in$  Rel
      by (simp add: symcl-def)
    moreover assume Pred b
    ultimately show Pred a
      using B2
    by blast
  qed
  from respection B[where Rel=Rel] show rel-respects-pred (symcl Rel) Pred
  by blast
have C:  $\bigwedge Rel. \text{rel-respects-pred } Rel \text{ Pred} \implies \text{rel-respects-pred } (Rel^+) \text{ Pred}$ 
proof -
  fix Rel
  assume rel-respects-pred Rel Pred
  thus rel-respects-pred (Rel+) Pred
    using preservation-and-closures(2)[where Rel=Rel and Pred=Pred]
      reflection-and-closures(2)[where Rel=Rel and Pred=Pred]
  by blast
  qed
  from respection C[where Rel=Rel] show rel-respects-pred (Rel+) Pred
  by blast
  from A B[where Rel=Rel=] show rel-respects-pred (symcl (Rel=)) Pred
  by blast
  from A C[where Rel=Rel=] show rel-respects-pred (Rel*) Pred
  using trancl-reflcl[of Rel]
  by fast
  from A B[where Rel=Rel=] C[where Rel=symcl (Rel=)]
  show rel-respects-pred ((symcl (Rel=))+) Pred
  by blast
qed

lemma respection-of-binary-predicates-and-closures:
  fixes Rel :: ('a  $\times$  'a) set
  and Pred :: 'a  $\Rightarrow$  'b  $\Rightarrow$  bool
  assumes respection: rel-respects-binary-pred Rel Pred
  shows rel-respects-binary-pred (Rel=) Pred

```

```

and rel-respects-binary-pred (symcl Rel) Pred
and rel-respects-binary-pred (Rel+) Pred
and rel-respects-binary-pred (symcl (Rel=)) Pred
and rel-respects-binary-pred (Rel*) Pred
and rel-respects-binary-pred ((symcl (Rel=))+) Pred
proof –
from respection show A: rel-respects-binary-pred (Rel=) Pred
  using preservation-of-binary-predicates-and-closures(1)[where Rel=Rel and Pred=Pred]
  reflection-of-binary-predicates-and-closures(1)[where Rel=Rel and Pred=Pred]
  by blast
have B:  $\bigwedge Rel. \text{rel-respects-binary-pred } Rel \text{ } Pred \implies \text{rel-respects-binary-pred } (symcl \text{ } Rel) \text{ } Pred$ 
proof
  fix Rel
  assume B1: rel-respects-binary-pred Rel Pred
  show rel-preserves-binary-pred (symcl Rel) Pred
  proof clarify
    fix a b x
    assume (a, b)  $\in$  symcl Rel
    hence (a, b)  $\in$  Rel  $\vee$  (b, a)  $\in$  Rel
    by (simp add: symcl-def)
    moreover assume Pred a x
    ultimately show Pred b x
    using B1
    by blast
  qed
next
  fix Rel
  assume B2: rel-respects-binary-pred Rel Pred
  show rel-reflects-binary-pred (symcl Rel) Pred
  proof clarify
    fix a b x
    assume (a, b)  $\in$  symcl Rel
    hence (a, b)  $\in$  Rel  $\vee$  (b, a)  $\in$  Rel
    by (simp add: symcl-def)
    moreover assume Pred b x
    ultimately show Pred a x
    using B2
    by blast
  qed
from respection B[where Rel=Rel] show rel-respects-binary-pred (symcl Rel) Pred
  by blast
have C:  $\bigwedge Rel. \text{rel-respects-binary-pred } Rel \text{ } Pred \implies \text{rel-respects-binary-pred } (Rel^+) \text{ } Pred$ 
proof –
  fix Rel
  assume rel-respects-binary-pred Rel Pred
  thus rel-respects-binary-pred (Rel+) Pred
  using preservation-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]
  reflection-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]
  by blast
qed
from respection C[where Rel=Rel] show rel-respects-binary-pred (Rel+) Pred
  by blast
from A B[where Rel=Rel=]
show rel-respects-binary-pred (symcl (Rel=)) Pred
  by blast
from A C[where Rel=Rel=]
show rel-respects-binary-pred (Rel*) Pred
  using trancl-reflcl[of Rel]
  by fast
from A B[where Rel=Rel=] C[where Rel=symcl (Rel=)]

```

```

show rel-respects-binary-pred ((symcl (Rel=))+) Pred
  by blast
qed

```

```

end
theory ProcessCalculi
  imports Relations
begin

```

## 2 Process Calculi

A process calculus is given by a set of process terms (syntax) and a relation on terms (semantics). We consider reduction as well as labelled variants of the semantics.

### 2.1 Reduction Semantics

A set of process terms and a relation on pairs of terms (called reduction semantics) define a process calculus.

```

record 'proc processCalculus =
  Reductions :: 'proc ⇒ 'proc ⇒ bool

```

A pair of the reduction relation is called a (reduction) step.

```

abbreviation step :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool
  (- ⟶ - - [70, 70, 70] 80)
  where
  P ⟶ Cal Q ≡ Reductions Cal P Q

```

We use  $*$  to indicate the reflexive and transitive closure of the reduction relation.

```

primrec nSteps
  :: 'proc ⇒ 'proc processCalculus ⇒ nat ⇒ 'proc ⇒ bool
  (- ⟶n - - [70, 70, 70, 70] 80)
  where
  P ⟶ Cal0 Q = (P = Q) |
  P ⟶ CalSuc n Q = (∃ P'. P ⟶ Caln P' ∧ P' ⟶ Cal Q)

```

```

definition steps
  :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool
  (- ⟶* - - [70, 70, 70] 80)
  where
  P ⟶ Cal* Q ≡ ∃ n. P ⟶ Caln Q

```

A process is divergent, if it can perform an infinite sequence of steps.

```

definition divergent
  :: 'proc ⇒ 'proc processCalculus ⇒ bool
  (- ⟶ω [70, 70] 80)
  where
  P ⟶ (Cal)ω ≡ ∀ P'. P ⟶ Cal* P' ⟶ (∃ P''. P' ⟶ Cal P'')

```

Each term can perform an (empty) sequence of steps to itself.

```

lemma steps-refl:
  fixes Cal :: 'proc processCalculus
  and P :: 'proc
  shows P ⟶ Cal* P
proof -
  have P ⟶ Cal0 P
  by simp
  hence ∃ n. P ⟶ Caln P

```

```

  by blast
  thus  $P \mapsto \text{Cal}^* P$ 
  by (simp add: steps-def)
qed

```

A single step is a sequence of steps of length one.

```

lemma step-to-steps:
  fixes  $\text{Cal} :: 'proc \text{ processCalculus}$ 
  and  $P P' :: 'proc$ 
  assumes  $\text{step}: P \mapsto \text{Cal} P'$ 
  shows  $P \mapsto \text{Cal}^* P'$ 
proof -
  from  $\text{step}$  have  $P \mapsto \text{Cal}^1 P'$ 
  by simp
  thus ?thesis
  unfolding steps-def
  by blast
qed

```

If there is a sequence of steps from P to Q and from Q to R, then there is also a sequence of steps from P to R.

```

lemma nSteps-add:
  fixes  $\text{Cal} :: 'proc \text{ processCalculus}$ 
  and  $n1 n2 :: nat$ 
  shows  $\forall P Q R. P \mapsto \text{Cal}^{n1} Q \wedge Q \mapsto \text{Cal}^{n2} R \longrightarrow P \mapsto \text{Cal}^{(n1 + n2)} R$ 
proof (induct n2, simp)
  case (Suc n)
  assume IH:  $\forall P Q R. P \mapsto \text{Cal}^{n1} Q \wedge Q \mapsto \text{Cal}^n R \longrightarrow P \mapsto \text{Cal}^{(n1 + n)} R$ 
  show ?case
  proof clarify
    fix  $P Q R$ 
    assume  $Q \mapsto \text{Cal}^{\text{Suc } n} R$ 
    from this obtain  $Q'$  where  $A1: Q \mapsto \text{Cal}^n Q'$  and  $A2: Q' \mapsto \text{Cal} R$ 
    by auto
    assume  $P \mapsto \text{Cal}^{n1} Q$ 
    with A1 IH have  $P \mapsto \text{Cal}^{(n1 + n)} Q'$ 
    by blast
    with A2 show  $P \mapsto \text{Cal}^{(n1 + \text{Suc } n)} R$ 
    by auto
  qed
qed

```

```

lemma steps-add:
  fixes  $\text{Cal} :: 'proc \text{ processCalculus}$ 
  and  $P Q R :: 'proc$ 
  assumes  $A1: P \mapsto \text{Cal}^* Q$ 
  and  $A2: Q \mapsto \text{Cal}^* R$ 
  shows  $P \mapsto \text{Cal}^* R$ 
proof -
  from A1 obtain  $n1$  where  $P \mapsto \text{Cal}^{n1} Q$ 
  by (auto simp add: steps-def)
  moreover from A2 obtain  $n2$  where  $Q \mapsto \text{Cal}^{n2} R$ 
  by (auto simp add: steps-def)
  ultimately have  $P \mapsto \text{Cal}^{(n1 + n2)} R$ 
  using nSteps-add[where  $\text{Cal}=\text{Cal}$ ]
  by blast
  thus  $P \mapsto \text{Cal}^* R$ 
  by (simp add: steps-def, blast)
qed

```

### 2.1.1 Observables or Barbs

We assume a predicate that tests terms for some kind of observables. At this point we do not limit or restrict the kind of observables used for a calculus nor the method to check them.

**record** (*'proc*, *'barbs*) *calculusWithBarbs* =  
*Calculus* :: *'proc* *processCalculus*  
*HasBarb* :: *'proc*  $\Rightarrow$  *'barbs*  $\Rightarrow$  *bool* ( $\dashv$ - [70, 70] 80)

**abbreviation** *hasBarb*  
:: *'proc*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *'barbs*  $\Rightarrow$  *bool*  
( $\dashv$ <->- [70, 70, 70] 80)  
**where**  
 $P \dashv \langle \text{CWB} \rangle a \equiv \text{HasBarb } \text{CWB } P a$

A term reaches a barb if it can evolve to a term that has this barb.

**abbreviation** *reachesBarb*  
:: *'proc*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *'barbs*  $\Rightarrow$  *bool*  
( $\dashv$ <->- [70, 70, 70] 80)  
**where**  
 $P \dashv \langle \text{CWB} \rangle a \equiv \exists P'. P \longmapsto (\text{Calculus } \text{CWB})^* P' \wedge P' \dashv \langle \text{CWB} \rangle a$

A relation R preserves barbs if whenever (P, Q) in R and P has a barb then also Q has this barb.

**abbreviation** *rel-preserves-barb-set*  
:: (*'proc*  $\times$  *'proc*) *set*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *'barbs set*  $\Rightarrow$  *bool*  
**where**  
*rel-preserves-barb-set* *Rel* *CWB* *Barbs*  $\equiv$   
*rel-preserves-binary-pred* *Rel* ( $\lambda P a. a \in \text{Barbs} \wedge P \dashv \langle \text{CWB} \rangle a$ )

**abbreviation** *rel-preserves-barbs*  
:: (*'proc*  $\times$  *'proc*) *set*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *bool*  
**where**  
*rel-preserves-barbs* *Rel* *CWB*  $\equiv$  *rel-preserves-binary-pred* *Rel* (*HasBarb* *CWB*)

**lemma** *preservation-of-barbs-and-set-of-barbs*:

**fixes** *Rel* :: (*'proc*  $\times$  *'proc*) *set*  
**and** *CWB* :: (*'proc*, *'barbs*) *calculusWithBarbs*  
**shows** *rel-preserves-barbs* *Rel* *CWB* = ( $\forall$  *Barbs*. *rel-preserves-barb-set* *Rel* *CWB* *Barbs*)  
**by** *blast*

A relation R reflects barbs if whenever (P, Q) in R and Q has a barb then also P has this barb.

**abbreviation** *rel-reflects-barb-set*  
:: (*'proc*  $\times$  *'proc*) *set*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *'barbs set*  $\Rightarrow$  *bool*  
**where**  
*rel-reflects-barb-set* *Rel* *CWB* *Barbs*  $\equiv$   
*rel-reflects-binary-pred* *Rel* ( $\lambda P a. a \in \text{Barbs} \wedge P \dashv \langle \text{CWB} \rangle a$ )

**abbreviation** *rel-reflects-barbs*  
:: (*'proc*  $\times$  *'proc*) *set*  $\Rightarrow$  (*'proc*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  *bool*  
**where**  
*rel-reflects-barbs* *Rel* *CWB*  $\equiv$  *rel-reflects-binary-pred* *Rel* (*HasBarb* *CWB*)

**lemma** *reflection-of-barbs-and-set-of-barbs*:

**fixes** *Rel* :: (*'proc*  $\times$  *'proc*) *set*  
**and** *CWB* :: (*'proc*, *'barbs*) *calculusWithBarbs*  
**shows** *rel-reflects-barbs* *Rel* *CWB* = ( $\forall$  *Barbs*. *rel-reflects-barb-set* *Rel* *CWB* *Barbs*)  
**by** *blast*

A relation respects barbs if it preserves and reflects barbs.

**abbreviation** *rel-respects-barb-set*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow 'barbs set \Rightarrow bool$   
**where**  
 $rel-respects-barb-set Rel CWB Barbs \equiv$   
 $rel-preserves-barb-set Rel CWB Barbs \wedge rel-reflects-barb-set Rel CWB Barbs$

**abbreviation** *rel-respects-barbs*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$   
**where**  
 $rel-respects-barbs Rel CWB \equiv rel-preserves-barbs Rel CWB \wedge rel-reflects-barbs Rel CWB$

**lemma** *respection-of-barbs-and-set-of-barbs:*

**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**shows**  $rel-respects-barbs Rel CWB = (\forall Barbs. rel-respects-barb-set Rel CWB Barbs)$   
**by** *blast*

If a relation preserves barbs then so does its reflexive or/and transitive closure.

**lemma** *preservation-of-barbs-and-closures:*

**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**assumes** *preservation: rel-preserves-barbs Rel CWB*  
**shows**  $rel-preserves-barbs (Rel^=) CWB$   
**and**  $rel-preserves-barbs (Rel^+) CWB$   
**and**  $rel-preserves-barbs (Rel^*) CWB$   
**using** *preservation*  
*preservation-of-binary-predicates-and-closures*[**where**  $Rel=Rel$  **and**  $Pred=HasBarb CWB$ ]  
**by** *blast+*

If a relation reflects barbs then so does its reflexive or/and transitive closure.

**lemma** *reflection-of-barbs-and-closures:*

**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**assumes** *reflection: rel-reflects-barbs Rel CWB*  
**shows**  $rel-reflects-barbs (Rel^=) CWB$   
**and**  $rel-reflects-barbs (Rel^+) CWB$   
**and**  $rel-reflects-barbs (Rel^*) CWB$   
**using** *reflection*  
*reflection-of-binary-predicates-and-closures*[**where**  $Rel=Rel$  **and**  $Pred=HasBarb CWB$ ]  
**by** *blast+*

If a relation respects barbs then so does its reflexive, symmetric, or/and transitive closure.

**lemma** *respection-of-barbs-and-closures:*

**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**assumes** *respection: rel-respects-barbs Rel CWB*  
**shows**  $rel-respects-barbs (Rel^=) CWB$   
**and**  $rel-respects-barbs (symcl Rel) CWB$   
**and**  $rel-respects-barbs (Rel^+) CWB$   
**and**  $rel-respects-barbs (symcl (Rel^=)) CWB$   
**and**  $rel-respects-barbs (Rel^*) CWB$   
**and**  $rel-respects-barbs ((symcl (Rel^=))^+) CWB$

**proof** –

**from** *respection* **show**  $rel-respects-barbs (Rel^=) CWB$   
**using** *respection-of-binary-predicates-and-closures*(1)[**where**  $Rel=Rel$  **and**  $Pred=HasBarb CWB$ ]  
**by** *blast*

**next**

**from** *respection* **show**  $rel-respects-barbs (symcl Rel) CWB$   
**using** *respection-of-binary-predicates-and-closures*(2)[**where**  $Rel=Rel$  **and**  $Pred=HasBarb CWB$ ]  
**by** *blast*

**next**

```

from respection show rel-respects-barbs ( $Rel^+$ ) CWB
  using respection-of-binary-predicates-and-closures(3)[where  $Rel=Rel$  and  $Pred=HasBarb$  CWB]
  by blast
next
from respection show rel-respects-barbs ( $symcl$  ( $Rel^=$ )) CWB
  using respection-of-binary-predicates-and-closures(4)[where  $Rel=Rel$  and  $Pred=HasBarb$  CWB]
  by blast
next
from respection show rel-respects-barbs ( $Rel^*$ ) CWB
  using respection-of-binary-predicates-and-closures(5)[where  $Rel=Rel$  and  $Pred=HasBarb$  CWB]
  by blast
next
from respection show rel-respects-barbs ( $(symcl$  ( $Rel^=$ )) $^+$ ) CWB
  using respection-of-binary-predicates-and-closures(6)[where  $Rel=Rel$  and  $Pred=HasBarb$  CWB]
  by blast
qed

```

A relation  $R$  weakly preserves barbs if it preserves reachability of barbs, i.e., if  $(P, Q)$  in  $R$  and  $P$  reaches a barb then also  $Q$  has to reach this barb.

**abbreviation** *rel-weakly-preserves-barb-set*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow 'barbs set \Rightarrow bool$   
**where**  
*rel-weakly-preserves-barb-set*  $Rel$  *CWB* *Barbs*  $\equiv$   
*rel-preserves-binary-pred*  $Rel$  ( $\lambda P a. a \in Barbs \wedge P \Downarrow \langle CWB \rangle a$ )

**abbreviation** *rel-weakly-preserves-barbs*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$   
**where**  
*rel-weakly-preserves-barbs*  $Rel$  *CWB*  $\equiv$  *rel-preserves-binary-pred*  $Rel$  ( $\lambda P a. P \Downarrow \langle CWB \rangle a$ )

**lemma** *weak-preservation-of-barbs-and-set-of-barbs*:  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**shows** *rel-weakly-preserves-barbs*  $Rel$  *CWB*  
 $= (\forall Barbs. rel-weakly-preserves-barb-set$   $Rel$  *CWB* *Barbs*)  
**by** *blast*

A relation  $R$  weakly reflects barbs if it reflects reachability of barbs, i.e., if  $(P, Q)$  in  $R$  and  $Q$  reaches a barb then also  $P$  has to reach this barb.

**abbreviation** *rel-weakly-reflects-barb-set*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow 'barbs set \Rightarrow bool$   
**where**  
*rel-weakly-reflects-barb-set*  $Rel$  *CWB* *Barbs*  $\equiv$   
*rel-reflects-binary-pred*  $Rel$  ( $\lambda P a. a \in Barbs \wedge P \Downarrow \langle CWB \rangle a$ )

**abbreviation** *rel-weakly-reflects-barbs*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$   
**where**  
*rel-weakly-reflects-barbs*  $Rel$  *CWB*  $\equiv$  *rel-reflects-binary-pred*  $Rel$  ( $\lambda P a. P \Downarrow \langle CWB \rangle a$ )

**lemma** *weak-reflection-of-barbs-and-set-of-barbs*:  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$   
**shows** *rel-weakly-reflects-barbs*  $Rel$  *CWB*  $= (\forall Barbs. rel-weakly-reflects-barb-set$   $Rel$  *CWB* *Barbs*)  
**by** *blast*

A relation weakly respects barbs if it weakly preserves and weakly reflects barbs.

**abbreviation** *rel-weakly-respects-barb-set*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow 'barbs set \Rightarrow bool$   
**where**

*rel-weakly-respects-barb-set Rel CWB Barbs*  $\equiv$   
*rel-weakly-preserves-barb-set Rel CWB Barbs*  $\wedge$  *rel-weakly-reflects-barb-set Rel CWB Barbs*

**abbreviation** *rel-weakly-respects-barbs*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$

**where**

*rel-weakly-respects-barbs Rel CWB*  $\equiv$

*rel-weakly-preserves-barbs Rel CWB*  $\wedge$  *rel-weakly-reflects-barbs Rel CWB*

**lemma** *weak-respection-of-barbs-and-set-of-barbs*:

**fixes** *Rel*  $:: ('proc \times 'proc) set$

**and** *CWB*  $:: ('proc, 'barbs) calculusWithBarbs$

**shows** *rel-weakly-respects-barbs Rel CWB*  $= (\forall Barbs. rel-weakly-respects-barb-set Rel CWB Barbs)$

**by** *blast*

If a relation weakly preserves barbs then so does its reflexive or/and transitive closure.

**lemma** *weak-preservation-of-barbs-and-closures*:

**fixes** *Rel*  $:: ('proc \times 'proc) set$

**and** *CWB*  $:: ('proc, 'barbs) calculusWithBarbs$

**assumes** *preservation*: *rel-weakly-preserves-barbs Rel CWB*

**shows** *rel-weakly-preserves-barbs (Rel<sup>=</sup>) CWB*

**and** *rel-weakly-preserves-barbs (Rel<sup>+</sup>) CWB*

**and** *rel-weakly-preserves-barbs (Rel<sup>\*</sup>) CWB*

**using** *preservation preservation-of-binary-predicates-and-closures*[**where** *Rel=Rel*

**and** *Pred* $=\lambda P a. P \Downarrow <CWB> a]$

**by** *blast+*

If a relation weakly reflects barbs then so does its reflexive or/and transitive closure.

**lemma** *weak-reflection-of-barbs-and-closures*:

**fixes** *Rel*  $:: ('proc \times 'proc) set$

**and** *CWB*  $:: ('proc, 'barbs) calculusWithBarbs$

**assumes** *reflection*: *rel-weakly-reflects-barbs Rel CWB*

**shows** *rel-weakly-reflects-barbs (Rel<sup>=</sup>) CWB*

**and** *rel-weakly-reflects-barbs (Rel<sup>+</sup>) CWB*

**and** *rel-weakly-reflects-barbs (Rel<sup>\*</sup>) CWB*

**using** *reflection reflection-of-binary-predicates-and-closures*[**where** *Rel=Rel*

**and** *Pred* $=\lambda P a. P \Downarrow <CWB> a]$

**by** *blast+*

If a relation weakly respects barbs then so does its reflexive, symmetric, or/and transitive closure.

**lemma** *weak-respection-of-barbs-and-closures*:

**fixes** *Rel*  $:: ('proc \times 'proc) set$

**and** *CWB*  $:: ('proc, 'barbs) calculusWithBarbs$

**assumes** *respection*: *rel-weakly-respects-barbs Rel CWB*

**shows** *rel-weakly-respects-barbs (Rel<sup>=</sup>) CWB*

**and** *rel-weakly-respects-barbs (symcl Rel) CWB*

**and** *rel-weakly-respects-barbs (Rel<sup>+</sup>) CWB*

**and** *rel-weakly-respects-barbs (symcl (Rel<sup>=</sup>)) CWB*

**and** *rel-weakly-respects-barbs (Rel<sup>\*</sup>) CWB*

**and** *rel-weakly-respects-barbs ((symcl (Rel<sup>=</sup>))<sup>+</sup>) CWB*

**proof** –

**from** *respection* **show** *rel-weakly-respects-barbs (Rel<sup>=</sup>) CWB*

**using** *respection-of-binary-predicates-and-closures*(1)[**where** *Rel=Rel*

**and** *Pred* $=\lambda P a. P \Downarrow <CWB> a]$

**by** *blast*

**next**

**from** *respection* **show** *rel-weakly-respects-barbs (symcl Rel) CWB*

**using** *respection-of-binary-predicates-and-closures*(2)[**where** *Rel=Rel*

**and** *Pred* $=\lambda P a. P \Downarrow <CWB> a]$

**by** *blast*



```

next
  from respection show rel-weakly-respects-barbs (Rel+) CWB
    using respection-of-binary-predicates-and-closures(3)[where Rel=Rel
      and Pred= $\lambda P a. P \Downarrow \langle CWB \rangle a$ ]
    by blast
next
  from respection show rel-weakly-respects-barbs (symcl (Rel=)) CWB
    using respection-of-binary-predicates-and-closures(4)[where Rel=Rel
      and Pred= $\lambda P a. P \Downarrow \langle CWB \rangle a$ ]
    by blast
next
  from respection show rel-weakly-respects-barbs (Rel*) CWB
    using respection-of-binary-predicates-and-closures(5)[where Rel=Rel
      and Pred= $\lambda P a. P \Downarrow \langle CWB \rangle a$ ]
    by blast
next
  from respection show rel-weakly-respects-barbs ((symcl (Rel=))+) CWB
    using respection-of-binary-predicates-and-closures(6)[where Rel=Rel
      and Pred= $\lambda P a. P \Downarrow \langle CWB \rangle a$ ]
    by blast
qed

end
theory SimulationRelations
  imports ProcessCalculi
begin

```

### 3 Simulation Relations

Simulation relations are a special kind of property on relations on processes. They usually require that steps are (strongly or weakly) preserved and/or reflected modulo the relation. We consider different kinds of simulation relations.

#### 3.1 Simulation

A weak reduction simulation is relation  $R$  such that if  $(P, Q)$  in  $R$  and  $P$  evolves to some  $P'$  then there exists some  $Q'$  such that  $Q$  evolves to  $Q'$  and  $(P', Q')$  in  $R$ .

**abbreviation** *weak-reduction-simulation*  
 $:: ('proc \times 'proc) set \Rightarrow 'proc \text{ processCalculus} \Rightarrow bool$   
**where**  
*weak-reduction-simulation*  $Rel \text{ Cal} \equiv$   
 $\forall P Q P'. (P, Q) \in Rel \wedge P \longmapsto \text{Cal} * P' \longrightarrow (\exists Q'. Q \longmapsto \text{Cal} * Q' \wedge (P', Q') \in Rel)$

A weak barbed simulation is weak reduction simulation that weakly preserves barbs.

**abbreviation** *weak-barbed-simulation*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) \text{ calculusWithBarbs} \Rightarrow bool$   
**where**  
*weak-barbed-simulation*  $Rel \text{ CWB} \equiv$   
*weak-reduction-simulation*  $Rel (\text{Calculus } CWB) \wedge \text{rel-weakly-preserves-barbs } Rel \text{ CWB}$

The reflexive and/or transitive closure of a weak simulation is a weak simulation.

**lemma** *weak-reduction-simulation-and-closures*:  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $Cal :: 'proc \text{ processCalculus}$   
**assumes** *simulation: weak-reduction-simulation*  $Rel \text{ Cal}$   
**shows** *weak-reduction-simulation*  $(Rel^=) \text{ Cal}$   
**and** *weak-reduction-simulation*  $(Rel^+) \text{ Cal}$   
**and** *weak-reduction-simulation*  $(Rel^*) \text{ Cal}$

**proof** –

**from** *simulation* **show**  $A$ : *weak-reduction-simulation*  $(Rel^=)$   $Cal$

**by** (*auto simp add: refl, blast*)

**have**  $B$ :  $\bigwedge Rel.$  *weak-reduction-simulation*  $Rel$   $Cal \implies$  *weak-reduction-simulation*  $(Rel^+)$   $Cal$

**proof** *clarify*

**fix**  $Rel$   $P$   $Q$   $P'$

**assume**  $B1$ : *weak-reduction-simulation*  $Rel$   $Cal$

**assume**  $(P, Q) \in Rel^+$  **and**  $P \mapsto_{Cal^*} P'$

**thus**  $\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^+$

**proof** (*induct arbitrary: P'*)

**fix**  $Q$   $P'$

**assume**  $(P, Q) \in Rel$  **and**  $P \mapsto_{Cal^*} P'$

**with**  $B1$  **obtain**  $Q'$  **where**  $Q \mapsto_{Cal^*} Q'$  **and**  $(P', Q') \in Rel$

**by** *blast*

**thus**  $\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^+$

**by** *auto*

**next**

**case** (*step*  $Q$   $R$   $P'$ )

**assume**  $\bigwedge P'. P \mapsto_{Cal^*} P' \implies (\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^+)$

**and**  $P \mapsto_{Cal^*} P'$

**from** *this* **obtain**  $Q'$  **where**  $B2$ :  $Q \mapsto_{Cal^*} Q'$  **and**  $B3$ :  $(P', Q') \in Rel^+$

**by** *blast*

**assume**  $(Q, R) \in Rel$

**with**  $B1$   $B2$  **obtain**  $R'$  **where**  $B4$ :  $R \mapsto_{Cal^*} R'$  **and**  $B5$ :  $(Q', R') \in Rel^+$

**by** *blast*

**from**  $B3$   $B5$  **have**  $(P', R') \in Rel^+$

**by** *simp*

**from**  $B4$  *this* **show**  $\exists R'. R \mapsto_{Cal^*} R' \wedge (P', R') \in Rel^+$

**by** *blast*

**qed**

**qed**

**with** *simulation* **show** *weak-reduction-simulation*  $(Rel^+)$   $Cal$

**by** *blast*

**from** *simulation*  $A$   $B$  [**where**  $Rel=Rel^=$ ]

**show** *weak-reduction-simulation*  $(Rel^*)$   $Cal$

**using** *trancl-reflcl*[*of Rel*]

**by** *fast*

**qed**

**lemma** *weak-barbed-simulation-and-closures*:

**fixes**  $Rel$  ::  $('proc \times 'proc)$  *set*

**and**  $CWB$  ::  $('proc, 'barbs)$  *calculusWithBarbs*

**assumes** *simulation*: *weak-barbed-simulation*  $Rel$   $CWB$

**shows** *weak-barbed-simulation*  $(Rel^=)$   $CWB$

**and** *weak-barbed-simulation*  $(Rel^+)$   $CWB$

**and** *weak-barbed-simulation*  $(Rel^*)$   $CWB$

**proof** –

**from** *simulation* **show** *weak-barbed-simulation*  $(Rel^=)$   $CWB$

**using** *weak-reduction-simulation-and-closures*(1)[**where**  $Rel=Rel$  **and**  $Cal=Calculus$   $CWB$ ]

*weak-preservation-of-barbs-and-closures*(1)[**where**  $Rel=Rel$  **and**  $CWB=CWB$ ]

**by** *blast*

**next**

**from** *simulation* **show** *weak-barbed-simulation*  $(Rel^+)$   $CWB$

**using** *weak-reduction-simulation-and-closures*(2)[**where**  $Rel=Rel$  **and**  $Cal=Calculus$   $CWB$ ]

*weak-preservation-of-barbs-and-closures*(2)[**where**  $Rel=Rel$  **and**  $CWB=CWB$ ]

**by** *blast*

**next**

**from** *simulation* **show** *weak-barbed-simulation*  $(Rel^*)$   $CWB$

**using** *weak-reduction-simulation-and-closures*(3)[**where**  $Rel=Rel$  **and**  $Cal=Calculus$   $CWB$ ]

*weak-preservation-of-barbs-and-closures*(3)[**where**  $Rel=Rel$  **and**  $CWB=CWB$ ]

**by** *blast*

qed

In the case of a simulation weak preservation of barbs can be replaced by the weaker condition that whenever  $(P, Q)$  in the relation and  $P$  has a barb then  $Q$  have to be able to reach this barb.

**abbreviation** *weak-barbed-preservation-cond*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$

**where**

$weak\text{-barbed-preservation-cond } Rel\ CWB \equiv \forall P\ Q\ a. (P, Q) \in Rel \wedge P \Downarrow \langle CWB \rangle a \longrightarrow Q \Downarrow \langle CWB \rangle a$

**lemma** *weak-preservation-of-barbs*:

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$

**assumes** *preservation: rel-weakly-preserves-barbs*  $Rel\ CWB$

**shows** *weak-barbed-preservation-cond*  $Rel\ CWB$

**proof** *clarify*

**fix**  $P\ Q\ a$

**have**  $P \mapsto (Calculus\ CWB)^* P$

**by** (*simp add: steps-refl*)

**moreover assume**  $P \Downarrow \langle CWB \rangle a$

**ultimately have**  $P \Downarrow \langle CWB \rangle a$

**by** *blast*

**moreover assume**  $(P, Q) \in Rel$

**ultimately show**  $Q \Downarrow \langle CWB \rangle a$

**using** *preservation*

**by** *blast*

qed

**lemma** *simulation-impl-equality-of-preservation-of-barbs-conditions*:

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$

**assumes** *simulation: weak-reduction-simulation*  $Rel\ (Calculus\ CWB)$

**shows** *rel-weakly-preserves-barbs*  $Rel\ CWB = weak\text{-barbed-preservation-cond } Rel\ CWB$

**proof**

**assume** *rel-weakly-preserves-barbs*  $Rel\ CWB$

**thus** *weak-barbed-preservation-cond*  $Rel\ CWB$

**using** *weak-preservation-of-barbs*[**where**  $Rel=Rel$  **and**  $CWB=CWB$ ]

**by** *blast*

**next**

**assume** *condition: weak-barbed-preservation-cond*  $Rel\ CWB$

**show** *rel-weakly-preserves-barbs*  $Rel\ CWB$

**proof** *clarify*

**fix**  $P\ Q\ a\ P'$

**assume**  $(P, Q) \in Rel$  **and**  $P \mapsto (Calculus\ CWB)^* P'$

**with** *simulation* **obtain**  $Q'$  **where**  $A1: Q \mapsto (Calculus\ CWB)^* Q'$  **and**  $A2: (P', Q') \in Rel$

**by** *blast*

**assume**  $P' \Downarrow \langle CWB \rangle a$

**with**  $A2$  *condition* **obtain**  $Q''$  **where**  $A3: Q' \mapsto (Calculus\ CWB)^* Q''$  **and**  $A4: Q'' \Downarrow \langle CWB \rangle a$

**by** *blast*

**from**  $A1\ A3$  **have**  $Q \mapsto (Calculus\ CWB)^* Q''$

**by** (*rule steps-add*)

**with**  $A4$  **show**  $Q \Downarrow \langle CWB \rangle a$

**by** *blast*

qed

qed

A strong reduction simulation is relation  $R$  such that for each pair  $(P, Q)$  in  $R$  and each step of  $P$  to some  $P'$  there exists some  $Q'$  such that there is a step of  $Q$  to  $Q'$  and  $(P', Q')$  in  $R$ .

**abbreviation** *strong-reduction-simulation*  $:: ('proc \times 'proc) set \Rightarrow 'proc\ processCalculus \Rightarrow bool$

**where**

$strong\text{-reduction-simulation } Rel\ Cal \equiv$

$\forall P Q P'. (P, Q) \in Rel \wedge P \mapsto Cal P' \longrightarrow (\exists Q'. Q \mapsto Cal Q' \wedge (P', Q') \in Rel)$

A strong barbed simulation is strong reduction simulation that preserves barbs.

**abbreviation** *strong-barbed-simulation*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$

**where**

*strong-barbed-simulation*  $Rel\ CWB \equiv$

*strong-reduction-simulation*  $Rel (Calculus\ CWB) \wedge rel\text{-preserves-barbs}\ Rel\ CWB$

A strong strong simulation is also a weak simulation.

**lemma** *strong-impl-weak-reduction-simulation*:

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $Cal :: 'proc\ processCalculus$

**assumes** *simulation*: *strong-reduction-simulation*  $Rel\ Cal$

**shows** *weak-reduction-simulation*  $Rel\ Cal$

**proof** *clarify*

**fix**  $P\ Q\ P'$

**assume**  $A1: (P, Q) \in Rel$

**assume**  $P \mapsto Cal^* P'$

**from** *this* **obtain**  $n$  **where**  $P \mapsto Cal^n P'$

**by** (*auto simp add: steps-def*)

**thus**  $\exists Q'. Q \mapsto Cal^* Q' \wedge (P', Q') \in Rel$

**proof** (*induct n arbitrary: P'*)

**case**  $0$

**assume**  $P \mapsto Cal^0 P'$

**hence**  $P = P'$

**by** (*simp add: steps-refl*)

**moreover** **have**  $Q \mapsto Cal^* Q$

**by** (*rule steps-refl*)

**ultimately** **show**  $\exists Q'. Q \mapsto Cal^* Q' \wedge (P', Q') \in Rel$

**using**  $A1$

**by** *blast*

**next**

**case** (*Suc n P''*)

**assume**  $P \mapsto Cal^{Suc\ n} P''$

**from** *this* **obtain**  $P'$  **where**  $A2: P \mapsto Cal^n P'$  **and**  $A3: P' \mapsto Cal P''$

**by** *auto*

**assume**  $\bigwedge P'. P \mapsto Cal^n P' \Longrightarrow \exists Q'. Q \mapsto Cal^* Q' \wedge (P', Q') \in Rel$

**with**  $A2$  **obtain**  $Q'$  **where**  $A4: Q \mapsto Cal^* Q'$  **and**  $A5: (P', Q') \in Rel$

**by** *blast*

**from** *simulation A5 A3* **obtain**  $Q''$  **where**  $A6: Q' \mapsto Cal Q''$  **and**  $A7: (P'', Q'') \in Rel$

**by** *blast*

**from**  $A4\ A6$  **have**  $Q \mapsto Cal^* Q''$

**using** *steps-add[where P=Q and Q=Q' and R=Q'']*

**by** (*simp add: step-to-steps*)

**with**  $A7$  **show**  $\exists Q'. Q \mapsto Cal^* Q' \wedge (P'', Q'') \in Rel$

**by** *blast*

**qed**

**qed**

**lemma** *strong-barbed-simulation-impl-weak-preservation-of-barbs*:

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $CWB :: ('proc, 'barbs) calculusWithBarbs$

**assumes** *simulation*: *strong-barbed-simulation*  $Rel\ CWB$

**shows** *rel-weakly-preserves-barbs*  $Rel\ CWB$

**proof** *clarify*

**fix**  $P\ Q\ a\ P'$

**assume**  $(P, Q) \in Rel$  **and**  $P \mapsto (Calculus\ CWB)^* P'$

**with** *simulation* **obtain**  $Q'$  **where**  $A1: Q \mapsto (Calculus\ CWB)^* Q'$  **and**  $A2: (P', Q') \in Rel$

**using** *strong-impl-weak-reduction-simulation[where Rel=Rel and Cal=Calculus CWB]*

```

  by blast
assume P'↓<CWB>a
with simulation A2 have Q'↓<CWB>a
  by blast
with A1 show Q↓<CWB>a
  by blast
qed

```

```

lemma strong-impl-weak-barbed-simulation:
fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes simulation: strong-barbed-simulation Rel CWB
shows weak-barbed-simulation Rel CWB
  using simulation
    strong-impl-weak-reduction-simulation[where Rel=Rel and Cal=Calculus CWB]
    strong-barbed-simulation-impl-weak-preservation-of-barbs[where Rel=Rel and CWB=CWB]
  by blast

```

The reflexive and/or transitive closure of a strong simulation is a strong simulation.

```

lemma strong-reduction-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
assumes simulation: strong-reduction-simulation Rel Cal
shows strong-reduction-simulation (Rel=) Cal
  and strong-reduction-simulation (Rel+) Cal
  and strong-reduction-simulation (Rel*) Cal
proof -
from simulation show A: strong-reduction-simulation (Rel=) Cal
  by (auto simp add: refl, blast)
have B:  $\bigwedge Rel.$  strong-reduction-simulation Rel Cal
   $\implies$  strong-reduction-simulation (Rel+) Cal
proof clarify
fix Rel P Q P'
assume B1: strong-reduction-simulation Rel Cal
assume (P, Q)  $\in$  Rel+ and P  $\mapsto$  Cal P'
thus  $\exists Q'. Q \mapsto Cal Q' \wedge (P', Q') \in Rel^+$ 
proof (induct arbitrary: P')
fix Q P'
assume (P, Q)  $\in$  Rel and P  $\mapsto$  Cal P'
with B1 obtain Q' where Q  $\mapsto$  Cal Q' and (P', Q')  $\in$  Rel
  by blast
thus  $\exists Q'. Q \mapsto Cal Q' \wedge (P', Q') \in Rel^+$ 
  by auto
next
case (step Q R P')
assume  $\bigwedge P'. P \mapsto Cal P' \implies (\exists Q'. Q \mapsto Cal Q' \wedge (P', Q') \in Rel^+)$ 
  and P  $\mapsto$  Cal P'
from this obtain Q' where B2: Q  $\mapsto$  Cal Q' and B3: (P', Q')  $\in$  Rel+
  by blast
assume (Q, R)  $\in$  Rel
with B1 B2 obtain R' where B4: R  $\mapsto$  Cal R' and B5: (Q', R')  $\in$  Rel+
  by blast
from B3 B5 have (P', R')  $\in$  Rel+
  by simp
with B4 show  $\exists R'. R \mapsto Cal R' \wedge (P', R') \in Rel^+$ 
  by blast
qed
qed
with simulation show strong-reduction-simulation (Rel+) Cal
  by blast
from simulation A B[where Rel=Rel=]

```

```

show strong-reduction-simulation (Rel*) Cal
  using trancl-reflcl[of Rel]
  by fast
qed

```

```

lemma strong-barbed-simulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: strong-barbed-simulation Rel CWB
  shows strong-barbed-simulation (Rel=) CWB
    and strong-barbed-simulation (Rel+) CWB
    and strong-barbed-simulation (Rel*) CWB
proof –
  from simulation show strong-barbed-simulation (Rel=) CWB
    using strong-reduction-simulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
      preservation-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
    by blast
next
  from simulation show strong-barbed-simulation (Rel+) CWB
    using strong-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
      preservation-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
    by blast
next
  from simulation show strong-barbed-simulation (Rel*) CWB
    using strong-reduction-simulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
      preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
    by blast
qed

```

### 3.2 Contrsimulation

A weak reduction contrsimulation is relation R such that if (P, Q) in R and P evolves to some P' then there exists some Q' such that Q evolves to Q' and (Q', P') in R.

```

abbreviation weak-reduction-contrsimulation
  :: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
  where
    weak-reduction-contrsimulation Rel Cal ≡
      ∀ P Q P'. (P, Q) ∈ Rel ∧ P ⟶ Cal* P' ⟶ (∃ Q'. Q ⟶ Cal* Q' ∧ (Q', P') ∈ Rel)

```

A weak barbed contrsimulation is weak reduction contrsimulation that weakly preserves barbs.

```

abbreviation weak-barbed-contrsimulation
  :: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
  where
    weak-barbed-contrsimulation Rel CWB ≡
      weak-reduction-contrsimulation Rel (Calculus CWB) ∧ rel-weakly-preserves-barbs Rel CWB

```

The reflexive and/or transitive closure of a weak contrsimulation is a weak contrsimulation.

```

lemma weak-reduction-contrsimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
    and Cal :: 'proc processCalculus
  assumes contrsimulation: weak-reduction-contrsimulation Rel Cal
  shows weak-reduction-contrsimulation (Rel=) Cal
    and weak-reduction-contrsimulation (Rel+) Cal
    and weak-reduction-contrsimulation (Rel*) Cal
proof –
  from contrsimulation show A: weak-reduction-contrsimulation (Rel=) Cal
    by (auto simp add: refl, blast)
  have B: ⋀ Rel. weak-reduction-contrsimulation Rel Cal
    ⟹ weak-reduction-contrsimulation (Rel+) Cal

```

```

proof clarify
  fix Rel P Q P'
  assume B1: weak-reduction-contrasimulation Rel Cal
  assume  $(P, Q) \in \text{Rel}^+$  and  $P \mapsto_{\text{Cal}^*} P'$ 
  thus  $\exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge (Q', P') \in \text{Rel}^+$ 
  proof (induct arbitrary: P')
    fix Q P'
    assume  $(P, Q) \in \text{Rel}$  and  $P \mapsto_{\text{Cal}^*} P'$ 
    with B1 obtain Q' where  $Q \mapsto_{\text{Cal}^*} Q'$  and  $(Q', P') \in \text{Rel}$ 
      by blast
    thus  $\exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge (Q', P') \in \text{Rel}^+$ 
      by auto
  next
  case (step Q R P')
  assume  $\bigwedge P'. P \mapsto_{\text{Cal}^*} P' \implies (\exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge (Q', P') \in \text{Rel}^+)$ 
    and  $P \mapsto_{\text{Cal}^*} P'$ 
  from this obtain Q' where B2: Q  $\mapsto_{\text{Cal}^*} Q'$  and B3: (Q', P')  $\in \text{Rel}^+$ 
    by blast
  assume  $(Q, R) \in \text{Rel}$ 
  with B1 B2 obtain R' where B4: R  $\mapsto_{\text{Cal}^*} R'$  and B5: (R', Q')  $\in \text{Rel}^+$ 
    by blast
  from B5 B3 have  $(R', P') \in \text{Rel}^+$ 
    by simp
  with B4 show  $\exists R'. R \mapsto_{\text{Cal}^*} R' \wedge (R', P') \in \text{Rel}^+$ 
    by blast
  qed
qed
with contrasimulation show weak-reduction-contrasimulation (Rel+) Cal
  by blast
from contrasimulation A B [where Rel=Rel=]
show weak-reduction-contrasimulation (Rel*) Cal
  using trancl-reflcl [of Rel]
  by fast
qed

```

**lemma** *weak-barbed-contrasimulation-and-closures:*

```

fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes contrasimulation: weak-barbed-contrasimulation Rel CWB
shows weak-barbed-contrasimulation (Rel=) CWB
  and weak-barbed-contrasimulation (Rel+) CWB
  and weak-barbed-contrasimulation (Rel*) CWB

```

**proof** –

```

from contrasimulation show weak-barbed-contrasimulation (Rel=) CWB
  using weak-reduction-contrasimulation-and-closures(1) [where Rel=Rel and Cal=Calculus CWB]
  weak-preservation-of-barbs-and-closures(1) [where Rel=Rel and CWB=CWB]
  by blast

```

**next**

```

from contrasimulation show weak-barbed-contrasimulation (Rel+) CWB
  using weak-reduction-contrasimulation-and-closures(2) [where Rel=Rel and Cal=Calculus CWB]
  weak-preservation-of-barbs-and-closures(2) [where Rel=Rel and CWB=CWB]
  by blast

```

**next**

```

from contrasimulation show weak-barbed-contrasimulation (Rel*) CWB
  using weak-reduction-contrasimulation-and-closures(3) [where Rel=Rel and Cal=Calculus CWB]
  weak-preservation-of-barbs-and-closures(3) [where Rel=Rel and CWB=CWB]
  by blast

```

**qed**

### 3.3 Coupled Simulation

A weak reduction coupled simulation is relation  $R$  such that if  $(P, Q)$  in  $R$  and  $P$  evolves to some  $P'$  then there exists some  $Q'$  such that  $Q$  evolves to  $Q'$  and  $(P', Q')$  in  $R$  and there exists some  $Q''$  such that  $Q$  evolves to  $Q''$  and  $(Q'', P')$  in  $R$ .

**abbreviation** *weak-reduction-coupled-simulation*

$:: ('proc \times 'proc) set \Rightarrow 'proc \text{ processCalculus} \Rightarrow bool$

**where**

*weak-reduction-coupled-simulation*  $Rel \text{ Cal} \equiv$

$\forall P Q P'. (P, Q) \in Rel \wedge P \mapsto Cal* P'$

$\longrightarrow (\exists Q'. Q \mapsto Cal* Q' \wedge (P', Q') \in Rel) \wedge (\exists Q''. Q \mapsto Cal* Q'' \wedge (Q'', P') \in Rel)$

A weak barbed coupled simulation is weak reduction coupled simulation that weakly preserves barbs.

**abbreviation** *weak-barbed-coupled-simulation*

$:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) \text{ calculusWithBarbs} \Rightarrow bool$

**where**

*weak-barbed-coupled-simulation*  $Rel \text{ CWB} \equiv$

*weak-reduction-coupled-simulation*  $Rel \text{ (Calculus CWB)} \wedge \text{rel-weakly-preserves-barbs } Rel \text{ CWB}$

A weak coupled simulation combines the conditions on a weak simulation and a weak contrasimulation.

**lemma** *weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation:*

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $Cal :: 'proc \text{ processCalculus}$

**shows** *weak-reduction-coupled-simulation*  $Rel \text{ Cal}$

$= (\text{weak-reduction-simulation } Rel \text{ Cal} \wedge \text{weak-reduction-contrasimulation } Rel \text{ Cal})$

**by** *blast*

**lemma** *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation:*

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$

**shows** *weak-barbed-coupled-simulation*  $Rel \text{ CWB}$

$= (\text{weak-barbed-simulation } Rel \text{ CWB} \wedge \text{weak-barbed-contrasimulation } Rel \text{ CWB})$

**by** *blast*

The reflexive and/or transitive closure of a weak coupled simulation is a weak coupled simulation.

**lemma** *weak-reduction-coupled-simulation-and-closures:*

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $Cal :: 'proc \text{ processCalculus}$

**assumes** *coupledSimulation: weak-reduction-coupled-simulation*  $Rel \text{ Cal}$

**shows** *weak-reduction-coupled-simulation*  $(Rel^=) \text{ Cal}$

**and** *weak-reduction-coupled-simulation*  $(Rel^+) \text{ Cal}$

**and** *weak-reduction-coupled-simulation*  $(Rel^*) \text{ Cal}$

**using** *weak-reduction-simulation-and-closures* [**where**  $Rel=Rel$  **and**  $Cal=Cal$ ]

*weak-reduction-contrasimulation-and-closures* [**where**  $Rel=Rel$  **and**  $Cal=Cal$ ]

*weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation* [**where**  $Rel=Rel$

**and**  $Cal=Cal$ ]

*coupledSimulation*

**by** *auto*

**lemma** *weak-barbed-coupled-simulation-and-closures:*

**fixes**  $Rel :: ('proc \times 'proc) set$

**and**  $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$

**assumes** *coupledSimulation: weak-barbed-coupled-simulation*  $Rel \text{ CWB}$

**shows** *weak-barbed-coupled-simulation*  $(Rel^=) \text{ CWB}$

**and** *weak-barbed-coupled-simulation*  $(Rel^+) \text{ CWB}$

**and** *weak-barbed-coupled-simulation*  $(Rel^*) \text{ CWB}$

**proof** –

**from** *coupledSimulation* **show** *weak-barbed-coupled-simulation*  $(Rel^=) \text{ CWB}$

**using** *weak-reduction-coupled-simulation-and-closures* (1) [**where**  $Rel=Rel$



```

    and Cal=Calculus CWB]
    weak-preservation-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
  by blast
next
from coupledSimulation show weak-barbed-coupled-simulation (Rel+) CWB
using weak-reduction-coupled-simulation-and-closures(2)[where Rel=Rel
and Cal=Calculus CWB]
weak-preservation-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
by blast
next
from coupledSimulation show weak-barbed-coupled-simulation (Rel*) CWB
using weak-reduction-coupled-simulation-and-closures(3)[where Rel=Rel
and Cal=Calculus CWB]
weak-preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
by blast
qed

```

### 3.4 Correspondence Simulation

A weak reduction correspondence simulation is relation  $R$  such that (1) if  $(P, Q)$  in  $R$  and  $P$  evolves to some  $P'$  then there exists some  $Q'$  such that  $Q$  evolves to  $Q'$  and  $(P', Q')$  in  $R$ , and (2) if  $(P, Q)$  in  $R$  and  $P$  evolves to some  $P''$  then there exists some  $Q''$  such that  $P$  evolves to  $P''$  and  $Q$  evolves to  $Q''$  and  $(P'', Q'')$  in  $Rel$ .

**abbreviation** *weak-reduction-correspondence-simulation*  
 $:: ('proc \times 'proc) set \Rightarrow 'proc processCalculus \Rightarrow bool$   
**where**  
*weak-reduction-correspondence-simulation Rel Cal*  $\equiv$   
 $(\forall P Q P'. (P, Q) \in Rel \wedge P \mapsto Cal* P' \longrightarrow (\exists Q'. Q \mapsto Cal* Q' \wedge (P', Q') \in Rel))$   
 $\wedge (\forall P Q Q'. (P, Q) \in Rel \wedge Q \mapsto Cal* Q' \longrightarrow (\exists P'' Q''. P \mapsto Cal* P'' \wedge Q' \mapsto Cal* Q'' \wedge (P'', Q'') \in Rel))$

A weak barbed correspondence simulation is weak reduction correspondence simulation that weakly respects barbs.

**abbreviation** *weak-barbed-correspondence-simulation*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) calculusWithBarbs \Rightarrow bool$   
**where**  
*weak-barbed-correspondence-simulation Rel CWB*  $\equiv$   
*weak-reduction-correspondence-simulation Rel (Calculus CWB)*  
 $\wedge rel-weakly-respects-barbs Rel CWB$

For each weak correspondence simulation  $R$  there exists a weak coupled simulation that contains all pairs of  $R$  in both directions.

**inductive-set** *cSim-cs*  $:: ('proc \times 'proc) set \Rightarrow 'proc processCalculus \Rightarrow ('proc \times 'proc) set$   
**for** *Rel*  $:: ('proc \times 'proc) set$   
**and** *Cal*  $:: 'proc processCalculus$   
**where**  
*left*:  $\llbracket Q \mapsto Cal* Q'; (P', Q') \in Rel \rrbracket \Longrightarrow (P', Q) \in cSim-cs Rel Cal \mid$   
*right*:  $\llbracket P \mapsto Cal* P'; (Q, P) \in Rel \rrbracket \Longrightarrow (P', Q) \in cSim-cs Rel Cal \mid$   
*trans*:  $\llbracket (P, Q) \in cSim-cs Rel Cal; (Q, R) \in cSim-cs Rel Cal \rrbracket \Longrightarrow (P, R) \in cSim-cs Rel Cal$

**lemma** *weak-reduction-correspondence-simulation-impl-coupled-simulation*:

**fixes** *Rel*  $:: ('proc \times 'proc) set$   
**and** *Cal*  $:: 'proc processCalculus$   
**assumes** *corrSim*: *weak-reduction-correspondence-simulation Rel Cal*  
**shows** *weak-reduction-coupled-simulation (cSim-cs Rel Cal) Cal*  
**and**  $\forall P Q. (P, Q) \in Rel \longrightarrow (P, Q) \in cSim-cs Rel Cal \wedge (Q, P) \in cSim-cs Rel Cal$   
**proof** –  
**show** *weak-reduction-coupled-simulation (cSim-cs Rel Cal) Cal*  
**proof** (*rule allI, rule allI, rule allI, rule impI, erule conjE*)

**fix**  $P Q P'$   
**assume**  $(P, Q) \in cSim\text{-}cs\ Rel\ Cal$  **and**  $P \mapsto Cal* P'$   
**thus**  $(\exists Q'. Q \mapsto Cal* Q' \wedge (P', Q') \in cSim\text{-}cs\ Rel\ Cal)$   
 $\wedge (\exists Q'. Q \mapsto Cal* Q' \wedge (Q', P') \in cSim\text{-}cs\ Rel\ Cal)$   
**proof** (*induct arbitrary: P'*)  
**case** (*left Q Q' P*)  
**assume**  $(P, Q') \in Rel$  **and**  $P \mapsto Cal* P'$   
**with** *corrSim* **obtain**  $Q''$  **where**  $A1: Q' \mapsto Cal* Q''$  **and**  $A2: (P', Q'') \in Rel$   
**by** *blast*  
**assume**  $A3: Q \mapsto Cal* Q'$   
**from** *this*  $A1$  **have**  $A4: Q \mapsto Cal* Q''$   
**by** (*rule steps-add*[*where P=Q and Q=Q' and R=Q''*])  
**have**  $Q'' \mapsto Cal* Q''$   
**by** (*rule steps-refl*)  
**with**  $A2$  **have**  $A5: (Q'', P') \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*simp add: cSim-cs.right*)  
**from**  $A1 A2$  **have**  $(P', Q') \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*rule cSim-cs.left*)  
**with**  $A4 A5 A3$  **show** *?case*  
**by** *blast*  
**next**  
**case** (*right P P' Q P''*)  
**assume**  $P \mapsto Cal* P'$  **and**  $P' \mapsto Cal* P''$   
**hence**  $B1: P \mapsto Cal* P''$   
**by** (*rule steps-add*[*where P=P and Q=P' and R=P''*])  
**assume**  $B2: (Q, P) \in Rel$   
**with** *corrSim*  $B1$  **obtain**  $Q''' P'''$  **where**  $B3: Q \mapsto Cal* Q'''$  **and**  $B4: P'' \mapsto Cal* P'''$   
**and**  $B5: (Q''', P''') \in Rel$   
**by** *blast*  
**from**  $B4 B5$  **have**  $B6: (Q''', P'') \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*rule cSim-cs.left*)  
**have**  $B7: Q \mapsto Cal* Q$   
**by** (*rule steps-refl*)  
**from**  $B1 B2$  **have**  $(P'', Q) \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*rule cSim-cs.right*)  
**with**  $B3 B6 B7$  **show** *?case*  
**by** *blast*  
**next**  
**case** (*trans P Q R P'*)  
**assume**  $P \mapsto Cal* P'$   
**and**  $\bigwedge P'. P \mapsto Cal* P' \implies (\exists Q'. Q \mapsto Cal* Q' \wedge (P', Q') \in cSim\text{-}cs\ Rel\ Cal)$   
 $\wedge (\exists Q'. Q \mapsto Cal* Q' \wedge (Q', P') \in cSim\text{-}cs\ Rel\ Cal)$   
**from** *this* **obtain**  $Q1 Q2$  **where**  $C1: Q \mapsto Cal* Q1$  **and**  $C2: (Q1, P') \in cSim\text{-}cs\ Rel\ Cal$   
**and**  $C3: Q \mapsto Cal* Q2$  **and**  $C4: (P', Q2) \in cSim\text{-}cs\ Rel\ Cal$   
**by** *blast*  
**assume**  $C5: \bigwedge Q'. Q \mapsto Cal* Q' \implies (\exists R'. R \mapsto Cal* R' \wedge (Q', R') \in cSim\text{-}cs\ Rel\ Cal)$   
 $\wedge (\exists R'. R \mapsto Cal* R' \wedge (R', Q') \in cSim\text{-}cs\ Rel\ Cal)$   
**with**  $C1$  **obtain**  $R1$  **where**  $C6: R \mapsto Cal* R1$  **and**  $C7: (R1, Q1) \in cSim\text{-}cs\ Rel\ Cal$   
**by** *blast*  
**from**  $C7 C2$  **have**  $C8: (R1, P') \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*rule cSim-cs.trans*)  
**from**  $C3 C5$  **obtain**  $R2$  **where**  $C9: R \mapsto Cal* R2$  **and**  $C10: (Q2, R2) \in cSim\text{-}cs\ Rel\ Cal$   
**by** *blast*  
**from**  $C4 C10$  **have**  $(P', R2) \in cSim\text{-}cs\ Rel\ Cal$   
**by** (*rule cSim-cs.trans*)  
**with**  $C6 C8 C9$  **show** *?case*  
**by** *blast*  
**qed**  
**qed**  
**next**  
**show**  $\forall P Q. (P, Q) \in Rel \longrightarrow (P, Q) \in cSim\text{-}cs\ Rel\ Cal \wedge (Q, P) \in cSim\text{-}cs\ Rel\ Cal$

```

proof clarify
  fix P Q
  have Q  $\mapsto$  Cal* Q
    by (rule steps-refl)
  moreover assume (P, Q)  $\in$  Rel
  ultimately show (P, Q)  $\in$  cSim-cs Rel Cal  $\wedge$  (Q, P)  $\in$  cSim-cs Rel Cal
    by (simp add: cSim-cs.left cSim-cs.right)
qed
qed

lemma weak-barbed-correspondence-simulation-impl-coupled-simulation:
  fixes Rel :: ('proc  $\times$  'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes corrSim: weak-barbed-correspondence-simulation Rel CWB
  shows weak-barbed-coupled-simulation (cSim-cs Rel (Calculus CWB)) CWB
  and  $\forall P Q. (P, Q) \in \text{Rel} \longrightarrow (P, Q) \in \text{cSim-cs Rel (Calculus CWB)}$ 
     $\wedge (Q, P) \in \text{cSim-cs Rel (Calculus CWB)}$ 

proof -
  show weak-barbed-coupled-simulation (cSim-cs Rel (Calculus CWB)) CWB
  proof
    from corrSim
    show weak-reduction-coupled-simulation (cSim-cs Rel (Calculus CWB)) (Calculus CWB)
      using weak-reduction-correspondence-simulation-impl-coupled-simulation(1)[where Rel=Rel
        and Cal=Calculus CWB]
    by blast
  next
  show rel-weakly-preserves-barbs (cSim-cs Rel (Calculus CWB)) CWB
  proof clarify
    fix P Q a P'
    assume (P, Q)  $\in$  cSim-cs Rel (Calculus CWB) and P  $\mapsto$  (Calculus CWB)* P' and P' $\downarrow$ <CWB>a
    thus Q $\downarrow$ <CWB>a
    proof (induct arbitrary: P')
      case (left Q Q' P P')
      assume (P, Q')  $\in$  Rel and P  $\mapsto$  (Calculus CWB)* P' and P' $\downarrow$ <CWB>a
      with corrSim obtain Q'' where A1: Q'  $\mapsto$  (Calculus CWB)* Q'' and A2: Q'' $\downarrow$ <CWB>a
      by blast
      assume Q  $\mapsto$  (Calculus CWB)* Q'
      from this A1 have Q  $\mapsto$  (Calculus CWB)* Q''
      by (rule steps-add)
      with A2 show Q $\downarrow$ <CWB>a
      by blast
    next
    case (right P P' Q P'')
    assume (Q, P)  $\in$  Rel
    moreover assume P  $\mapsto$  (Calculus CWB)* P' and P'  $\mapsto$  (Calculus CWB)* P''
    hence P  $\mapsto$  (Calculus CWB)* P''
    by (rule steps-add)
    moreover assume P'' $\downarrow$ <CWB>a
    ultimately show Q $\downarrow$ <CWB>a
    using corrSim
    by blast
  next
  case (trans P Q R P')
  assume  $\bigwedge P'. P \mapsto (\text{Calculus CWB})^* P' \implies P' \downarrow \langle \text{CWB} \rangle a \implies Q \downarrow \langle \text{CWB} \rangle a$ 
    and P  $\mapsto$  (Calculus CWB)* P' and P' $\downarrow$ <CWB>a
    and  $\bigwedge Q'. Q \mapsto (\text{Calculus CWB})^* Q' \implies Q' \downarrow \langle \text{CWB} \rangle a \implies R \downarrow \langle \text{CWB} \rangle a$ 
  thus R $\downarrow$ <CWB>a
  by blast
  qed
qed
qed

```

**next**  
**from** *corrSim* **show**  $\forall P Q. (P, Q) \in \text{Rel} \longrightarrow (P, Q) \in \text{cSim-cs Rel (Calculus CWB)}$   
 $\quad \wedge (Q, P) \in \text{cSim-cs Rel (Calculus CWB)}$   
**using** *weak-reduction-correspondence-simulation-impl-coupled-simulation(2)* [**where**  $\text{Rel}=\text{Rel}$   
**and**  $\text{Cal}=\text{Calculus CWB}$ ]  
**by** *blast*  
**qed**

**lemma** *reduction-correspondence-simulation-condition-trans:*  
**fixes**  $\text{Cal} :: \text{'proc processCalculus}$   
**and**  $P Q R :: \text{'proc}$   
**and**  $\text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set}$   
**assumes**  $A1: \forall Q'. Q \longmapsto \text{Cal}^* Q' \longrightarrow (\exists P'' Q''. P \longmapsto \text{Cal}^* P'' \wedge Q' \longmapsto \text{Cal}^* Q'' \wedge (P'', Q'') \in \text{Rel})$   
**and**  $A2: \forall R'. R \longmapsto \text{Cal}^* R' \longrightarrow (\exists Q'' R''. Q \longmapsto \text{Cal}^* Q'' \wedge R' \longmapsto \text{Cal}^* R'' \wedge (Q'', R'') \in \text{Rel})$   
**and**  $A3: \text{weak-reduction-simulation Rel Cal}$   
**and**  $A4: \text{trans Rel}$   
**shows**  $\forall R'. R \longmapsto \text{Cal}^* R' \longrightarrow (\exists P'' R''. P \longmapsto \text{Cal}^* P'' \wedge R' \longmapsto \text{Cal}^* R'' \wedge (P'', R'') \in \text{Rel})$

**proof** *clarify*  
**fix**  $R'$   
**assume**  $R \longmapsto \text{Cal}^* R'$   
**with**  $A2$  **obtain**  $Q'' R''$  **where**  $A5: Q \longmapsto \text{Cal}^* Q''$  **and**  $A6: R' \longmapsto \text{Cal}^* R''$   
**and**  $A7: (Q'', R'') \in \text{Rel}$   
**by** *blast*  
**from**  $A1 A5$  **obtain**  $P''' Q'''$  **where**  $A8: P \longmapsto \text{Cal}^* P'''$  **and**  $A9: Q'' \longmapsto \text{Cal}^* Q'''$   
**and**  $A10: (P''', Q''') \in \text{Rel}$   
**by** *blast*  
**from**  $A3 A7 A9$  **obtain**  $R'''$  **where**  $A11: R'' \longmapsto \text{Cal}^* R'''$  **and**  $A12: (Q''', R''') \in \text{Rel}$   
**by** *blast*  
**from**  $A6 A11$  **have**  $A13: R' \longmapsto \text{Cal}^* R'''$   
**by** (*rule steps-add* [**where**  $P=R'$  **and**  $Q=R''$  **and**  $R=R'''$ ])  
**from**  $A4 A10 A12$  **have**  $(P''', R''') \in \text{Rel}$   
**unfolding** *trans-def*  
**by** *blast*  
**with**  $A8 A13$  **show**  $\exists P'' R''. P \longmapsto \text{Cal}^* P'' \wedge R' \longmapsto \text{Cal}^* R'' \wedge (P'', R'') \in \text{Rel}$   
**by** *blast*  
**qed**

The reflexive and/or transitive closure of a weak correspondence simulation is a weak correspondence simulation.

**lemma** *weak-reduction-correspondence-simulation-and-closures:*  
**fixes**  $\text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set}$   
**and**  $\text{Cal} :: \text{'proc processCalculus}$   
**assumes** *corrSim: weak-reduction-correspondence-simulation Rel Cal*  
**shows** *weak-reduction-correspondence-simulation (Rel<sup>=</sup>) Cal*  
**and** *weak-reduction-correspondence-simulation (Rel<sup>+</sup>) Cal*  
**and** *weak-reduction-correspondence-simulation (Rel<sup>\*</sup>) Cal*

**proof** –  
**show**  $A: \text{weak-reduction-correspondence-simulation (Rel}^{\text{=}}\text{) Cal}$   
**proof**  
**from** *corrSim* **show** *weak-reduction-simulation (Rel}^{\text{=}}\text{) Cal*  
**using** *weak-reduction-simulation-and-closures(1)* [**where**  $\text{Rel}=\text{Rel}$  **and**  $\text{Cal}=\text{Cal}$ ]  
**by** *blast*

**next**  
**show**  $\forall P Q Q'. (P, Q) \in \text{Rel}^{\text{=}} \wedge Q \longmapsto \text{Cal}^* Q'$   
 $\longrightarrow (\exists P'' Q''. P \longmapsto \text{Cal}^* P'' \wedge Q' \longmapsto \text{Cal}^* Q'' \wedge (P'', Q'') \in \text{Rel}^{\text{=}})$

**proof** *clarify*  
**fix**  $P Q Q'$   
**assume**  $(P, Q) \in \text{Rel}^{\text{=}}$  **and**  $A1: Q \longmapsto \text{Cal}^* Q'$   
**moreover** **have**  $P = Q \implies \exists P'' Q''. P \longmapsto \text{Cal}^* P'' \wedge Q' \longmapsto \text{Cal}^* Q'' \wedge (P'', Q'') \in \text{Rel}^{\text{=}}$   
**proof** –

```

assume P = Q
moreover have Q'  $\mapsto$  Cal* Q'
  by (rule steps-refl)
ultimately show  $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel=
  using A1 refl
  by blast
qed
moreover
have (P, Q)  $\in$  Rel  $\implies$   $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel=
proof -
  assume (P, Q)  $\in$  Rel
  with corrSim A1 obtain P'' Q'' where P  $\mapsto$  Cal* P'' and Q'  $\mapsto$  Cal* Q''
    and (P'', Q'')  $\in$  Rel
    by blast
  thus  $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel=
    by auto
qed
ultimately show  $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel=
  by auto
qed
have B:  $\bigwedge$  Rel. weak-reduction-correspondence-simulation Rel Cal
   $\implies$  weak-reduction-correspondence-simulation (Rel+) Cal
proof
  fix Rel
  assume weak-reduction-correspondence-simulation Rel Cal
  thus weak-reduction-simulation (Rel+) Cal
    using weak-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Cal]
    by blast
next
  fix Rel
  assume B1: weak-reduction-correspondence-simulation Rel Cal
  show  $\forall P Q Q'. (P, Q) \in$  Rel+  $\wedge$  Q  $\mapsto$  Cal* Q'
     $\longrightarrow$  ( $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel+)
  proof clarify
    fix P Q Q'
    assume (P, Q)  $\in$  Rel+ and Q  $\mapsto$  Cal* Q'
    thus  $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel+
  proof (induct arbitrary: Q')
    fix Q Q'
    assume (P, Q)  $\in$  Rel and Q  $\mapsto$  Cal* Q'
    with B1 obtain P'' Q'' where B2: P  $\mapsto$  Cal* P'' and B3: Q'  $\mapsto$  Cal* Q''
      and B4: (P'', Q'')  $\in$  Rel
      by blast
    from B4 have (P'', Q'')  $\in$  Rel+
      by simp
    with B2 B3 show  $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel+
      by blast
  next
  case (step Q R R')
  assume  $\bigwedge$  Q'. Q  $\mapsto$  Cal* Q'
     $\implies$   $\exists P'' Q''. P \mapsto$  Cal* P''  $\wedge$  Q'  $\mapsto$  Cal* Q''  $\wedge$  (P'', Q'')  $\in$  Rel+
  moreover assume (Q, R)  $\in$  Rel
  with B1
  have  $\bigwedge$  R'. R  $\mapsto$  Cal* R'  $\implies$   $\exists Q'' R''. Q \mapsto$  Cal* Q''  $\wedge$  R'  $\mapsto$  Cal* R''  $\wedge$  (Q'', R'')  $\in$  Rel+
    by blast
  moreover from B1 have weak-reduction-simulation (Rel+) Cal
    using weak-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Cal]
    by blast
  moreover have trans (Rel+)
    using trans-trancl[of Rel]

```

```

    by blast
    moreover assume  $R \mapsto \text{Cal} * R'$ 
    ultimately show  $\exists P'' R''. P \mapsto \text{Cal} * P'' \wedge R' \mapsto \text{Cal} * R'' \wedge (P'', R'') \in \text{Rel}^+$ 
      using reduction-correspondence-simulation-condition-trans[where  $\text{Rel} = \text{Rel}^+$ ]
    by blast
  qed
qed
qed
from corrSim B[where  $\text{Rel} = \text{Rel}$ ] show weak-reduction-correspondence-simulation  $(\text{Rel}^+)$  Cal
  by blast
from A B[where  $\text{Rel} = \text{Rel}^=$ ]
show weak-reduction-correspondence-simulation  $(\text{Rel}^*)$  Cal
  using trancl-reflcl[of Rel]
  by auto
qed

```

**lemma** *weak-barbed-correspondence-simulation-and-closures:*

```

fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes corrSim: weak-barbed-correspondence-simulation Rel CWB
shows weak-barbed-correspondence-simulation  $(\text{Rel}^=)$  CWB
  and weak-barbed-correspondence-simulation  $(\text{Rel}^+)$  CWB
  and weak-barbed-correspondence-simulation  $(\text{Rel}^*)$  CWB
proof -
  from corrSim show weak-barbed-correspondence-simulation  $(\text{Rel}^=)$  CWB
    using weak-reduction-correspondence-simulation-and-closures(1)[where  $\text{Rel} = \text{Rel}$ 
      and  $\text{Cal} = \text{Calculus CWB}$ ]
      weak-respection-of-barbs-and-closures(1)[where  $\text{Rel} = \text{Rel}$  and  $\text{CWB} = \text{CWB}$ ]
    by fast
next
  from corrSim show weak-barbed-correspondence-simulation  $(\text{Rel}^+)$  CWB
    using weak-reduction-correspondence-simulation-and-closures(2)[where  $\text{Rel} = \text{Rel}$ 
      and  $\text{Cal} = \text{Calculus CWB}$ ]
      weak-respection-of-barbs-and-closures(3)[where  $\text{Rel} = \text{Rel}$  and  $\text{CWB} = \text{CWB}$ ]
    by blast
next
  from corrSim show weak-barbed-correspondence-simulation  $(\text{Rel}^*)$  CWB
    using weak-reduction-correspondence-simulation-and-closures(3)[where  $\text{Rel} = \text{Rel}$ 
      and  $\text{Cal} = \text{Calculus CWB}$ ]
      weak-respection-of-barbs-and-closures(5)[where  $\text{Rel} = \text{Rel}$  and  $\text{CWB} = \text{CWB}$ ]
    by blast
qed

```

### 3.5 Bisimulation

A weak reduction bisimulation is relation  $R$  such that (1) if  $(P, Q)$  in  $R$  and  $P$  evolves to some  $P'$  then there exists some  $Q'$  such that  $Q$  evolves to  $Q'$  and  $(P', Q')$  in  $R$ , and (2) if  $(P, Q)$  in  $R$  and  $Q$  evolves to some  $Q'$  then there exists some  $P'$  such that  $P$  evolves to  $P'$  and  $(P', Q')$  in  $R$ .

**abbreviation** *weak-reduction-bisimulation*

```

:: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
where
  weak-reduction-bisimulation Rel Cal ≡
  (∀ P Q P'. (P, Q) ∈ Rel ∧ P ↦ Cal * P' → (∃ Q'. Q ↦ Cal * Q' ∧ (P', Q') ∈ Rel))
  ∧ (∀ P Q Q'. (P, Q) ∈ Rel ∧ Q ↦ Cal * Q' → (∃ P'. P ↦ Cal * P' ∧ (P', Q') ∈ Rel))

```

A weak barbed bisimulation is weak reduction bisimulation that weakly respects barbs.

**abbreviation** *weak-barbed-bisimulation*

```

:: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
where
  weak-barbed-bisimulation Rel CWB ≡

```

*weak-reduction-bisimulation Rel (Calculus CWB)  $\wedge$  rel-weakly-respects-barbs Rel CWB*

A symmetric weak simulation is a weak bisimulation.

**lemma** *symm-weak-reduction-simulation-is-bisimulation:*

**fixes** *Rel* :: ('proc  $\times$  'proc) set  
**and** *Cal* :: 'proc processCalculus  
**assumes** *sym Rel*  
**and** *weak-reduction-simulation Rel Cal*  
**shows** *weak-reduction-bisimulation Rel Cal*  
**using** *assms symD[of Rel]*  
**by** *blast*

**lemma** *symm-weak-barbed-simulation-is-bisimulation:*

**fixes** *Rel* :: ('proc  $\times$  'proc) set  
**and** *CWB* :: ('proc, 'barbs) calculusWithBarbs  
**assumes** *sym Rel*  
**and** *weak-barbed-simulation Rel Cal*  
**shows** *weak-barbed-bisimulation Rel Cal*  
**using** *assms symD[of Rel]*  
**by** *blast*

If a relation as well as its inverse are weak simulations, then this relation is a weak bisimulation.

**lemma** *weak-reduction-simulations-impl-bisimulation:*

**fixes** *Rel* :: ('proc  $\times$  'proc) set  
**and** *Cal* :: 'proc processCalculus  
**assumes** *sim*: *weak-reduction-simulation Rel Cal*  
**and** *simInv*: *weak-reduction-simulation (Rel<sup>-1</sup>) Cal*  
**shows** *weak-reduction-bisimulation Rel Cal*

**proof** *auto*

**fix** *P Q P'*  
**assume**  $(P, Q) \in Rel$  **and**  $P \mapsto_{Cal^*} P'$   
**with** *sim* **show**  $\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel$   
**by** *simp*

**next**

**fix** *P Q Q'*  
**assume**  $(P, Q) \in Rel$   
**hence**  $(Q, P) \in Rel^{-1}$   
**by** *simp*  
**moreover** **assume**  $Q \mapsto_{Cal^*} Q'$   
**ultimately obtain** *P'* **where** *A1*:  $P \mapsto_{Cal^*} P'$  **and** *A2*:  $(Q', P') \in Rel^{-1}$   
**using** *simInv*  
**by** *blast*  
**from** *A2* **have**  $(P', Q') \in Rel$   
**by** *induct*  
**with** *A1* **show**  $\exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q') \in Rel$   
**by** *blast*

**qed**

**lemma** *weak-reduction-bisimulations-impl-inverse-is-simulation:*

**fixes** *Rel* :: ('proc  $\times$  'proc) set  
**and** *Cal* :: 'proc processCalculus  
**assumes** *bisim*: *weak-reduction-bisimulation Rel Cal*  
**shows** *weak-reduction-simulation (Rel<sup>-1</sup>) Cal*

**proof** *clarify*

**fix** *P Q P'*  
**assume**  $(Q, P) \in Rel$   
**moreover** **assume**  $P \mapsto_{Cal^*} P'$   
**ultimately obtain** *Q'* **where** *A1*:  $Q \mapsto_{Cal^*} Q'$  **and** *A2*:  $(Q', P') \in Rel$   
**using** *bisim*  
**by** *blast*

```

from  $A2$  have  $(P', Q') \in Rel^{-1}$ 
  by simp
with  $A1$  show  $\exists Q'. Q \mapsto Cal * Q' \wedge (P', Q') \in Rel^{-1}$ 
  by blast
qed

```

**lemma** *weak-reduction-simulations-iff-bisimulation:*

```

fixes  $Rel :: ('proc \times 'proc)$  set
  and  $Cal :: 'proc$  processCalculus
shows  $(weak-reduction-simulation\ Rel\ Cal \wedge weak-reduction-simulation\ (Rel^{-1})\ Cal)$ 
   $= weak-reduction-bisimulation\ Rel\ Cal$ 
  using weak-reduction-simulations-impl-bisimulation[where  $Rel=Rel$  and  $Cal=Cal$ ]
  weak-reduction-bisimulations-impl-inverse-is-simulation[where  $Rel=Rel$  and  $Cal=Cal$ ]
  by blast

```

**lemma** *weak-barbed-simulations-iff-bisimulation:*

```

fixes  $Rel :: ('proc \times 'proc)$  set
  and  $CWB :: ('proc, 'barbs)$  calculusWithBarbs
shows  $(weak-barbed-simulation\ Rel\ CWB \wedge weak-barbed-simulation\ (Rel^{-1})\ CWB)$ 
   $= weak-barbed-bisimulation\ Rel\ CWB$ 

```

**proof** (*rule iffI, erule conjE*)

```

assume  $sim: weak-barbed-simulation\ Rel\ CWB$ 
  and  $rev: weak-barbed-simulation\ (Rel^{-1})\ CWB$ 
hence weak-reduction-bisimulation  $Rel$  (Calculus  $CWB$ )
  using weak-reduction-simulations-impl-bisimulation[where  $Rel=Rel$  and  $Cal=Calculus\ CWB$ ]
  by blast

```

**moreover from**  $sim$  **have** *rel-weakly-preserves-barbs*  $Rel\ CWB$

**by** *simp*

**moreover from**  $rev$  **have** *rel-weakly-reflects-barbs*  $Rel\ CWB$

**by** *simp*

**ultimately show** *weak-barbed-bisimulation*  $Rel\ CWB$

**by** *blast*

**next**

**assume**  $bisim: weak-barbed-bisimulation\ Rel\ CWB$

**hence** *weak-barbed-simulation*  $Rel\ CWB$

**by** *blast*

**moreover from**  $bisim$  **have** *weak-reduction-simulation*  $(Rel^{-1})$  (*Calculus*  $CWB$ )

**using** *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where**  $Rel=Rel$ ]

**by** *simp*

**moreover from**  $bisim$  **have** *rel-weakly-reflects-barbs*  $Rel\ CWB$

**by** *blast*

**hence** *rel-weakly-preserves-barbs*  $(Rel^{-1})\ CWB$

**by** *simp*

**ultimately show** *weak-barbed-simulation*  $Rel\ CWB \wedge weak-barbed-simulation\ (Rel^{-1})\ CWB$

**by** *blast*

**qed**

A weak bisimulation is a weak correspondence simulation.

**lemma** *weak-reduction-bisimulation-is-correspondence-simulation:*

```

fixes  $Rel :: ('proc \times 'proc)$  set

```

```

  and  $Cal :: 'proc$  processCalculus

```

```

assumes  $bisim: weak-reduction-bisimulation\ Rel\ Cal$ 

```

```

shows weak-reduction-correspondence-simulation  $Rel\ Cal$ 

```

**proof**

```

from  $bisim$  show weak-reduction-simulation  $Rel\ Cal$ 

```

**by** *blast*

**next**

```

show  $\forall P\ Q\ Q'. (P, Q) \in Rel \wedge Q \mapsto Cal * Q'$ 

```

```

   $\longrightarrow (\exists P''\ Q''. P \mapsto Cal * P'' \wedge Q' \mapsto Cal * Q'' \wedge (P'', Q'') \in Rel)$ 

```

**proof** *clarify*

```

  fix  $P\ Q\ Q'$ 

```



```

assume  $(P, Q) \in Rel$  and  $Q \mapsto_{Cal*} Q'$ 
with bisim obtain  $P'$  where  $P \mapsto_{Cal*} P'$  and  $(P', Q') \in Rel$ 
  by blast
moreover have  $Q' \mapsto_{Cal*} Q'$ 
  by (rule steps-refl)
ultimately show  $(\exists P'' Q''. P \mapsto_{Cal*} P'' \wedge Q' \mapsto_{Cal*} Q'' \wedge (P'', Q'') \in Rel)$ 
  by blast
qed
qed

```

**lemma** *weak-barbed-bisimulation-is-correspondence-simulation*:

```

fixes  $Rel :: ('proc \times 'proc) set$ 
  and  $CWB :: ('proc, 'barbs) calculusWithBarbs$ 
assumes bisim: weak-barbed-bisimulation  $Rel$   $CWB$ 
shows weak-barbed-correspondence-simulation  $Rel$   $CWB$ 
  using bisim weak-reduction-bisimulation-is-correspondence-simulation[where  $Rel=Rel$ 
    and  $Cal=Calculus$   $CWB$ ]
  by blast

```

The reflexive, symmetric, and/or transitive closure of a weak bisimulation is a weak bisimulation.

**lemma** *weak-reduction-bisimulation-and-closures*:

```

fixes  $Rel :: ('proc \times 'proc) set$ 
  and  $Cal :: 'proc processCalculus$ 
assumes bisim: weak-reduction-bisimulation  $Rel$   $Cal$ 
shows weak-reduction-bisimulation  $(Rel^=)$   $Cal$ 
  and weak-reduction-bisimulation  $(symcl\ Rel)$   $Cal$ 
  and weak-reduction-bisimulation  $(Rel^+)$   $Cal$ 
  and weak-reduction-bisimulation  $(symcl\ (Rel^=))$   $Cal$ 
  and weak-reduction-bisimulation  $(Rel^*)$   $Cal$ 
  and weak-reduction-bisimulation  $((symcl\ (Rel^=))^+)$   $Cal$ 
proof –
from bisim show  $A$ : weak-reduction-bisimulation  $(Rel^=)$   $Cal$ 
  by (auto simp add: refl, blast+)
have  $B$ :  $\bigwedge Rel.$  weak-reduction-bisimulation  $Rel$   $Cal$ 
   $\implies$  weak-reduction-bisimulation  $(symcl\ Rel)$   $Cal$ 
  by (auto simp add: symcl-def, blast+)
from bisim  $B$ [where  $Rel=Rel$ ] show weak-reduction-bisimulation  $(symcl\ Rel)$   $Cal$ 
  by blast
have  $C$ :  $\bigwedge Rel.$  weak-reduction-bisimulation  $Rel$   $Cal$ 
   $\implies$  weak-reduction-bisimulation  $(Rel^+)$   $Cal$ 
proof
  fix  $Rel$ 
  assume weak-reduction-bisimulation  $Rel$   $Cal$ 
  thus weak-reduction-simulation  $(Rel^+)$   $Cal$ 
    using weak-reduction-simulation-and-closures(2)[where  $Rel=Rel$  and  $Cal=Cal$ ]
    by blast
next
  fix  $Rel$ 
  assume  $C1$ : weak-reduction-bisimulation  $Rel$   $Cal$ 
  show  $\forall P\ Q\ Q'. (P, Q) \in Rel^+ \wedge Q \mapsto_{Cal*} Q'$ 
     $\longrightarrow (\exists P'. P \mapsto_{Cal*} P' \wedge (P', Q') \in Rel^+)$ 
  proof clarify
    fix  $P\ Q\ Q'$ 
    assume  $(P, Q) \in Rel^+$  and  $Q \mapsto_{Cal*} Q'$ 
    thus  $\exists P'. P \mapsto_{Cal*} P' \wedge (P', Q') \in Rel^+$ 
    proof (induct arbitrary: Q')
      fix  $Q\ Q'$ 
      assume  $(P, Q) \in Rel$  and  $Q \mapsto_{Cal*} Q'$ 
      with  $C1$  obtain  $P'$  where  $P \mapsto_{Cal*} P'$  and  $(P', Q') \in Rel$ 
      by blast
      thus  $\exists P'. P \mapsto_{Cal*} P' \wedge (P', Q') \in Rel^+$ 
    
```

```

    by auto
next
  case (step Q R R')
  assume (Q, R) ∈ Rel and R ↦ Cal* R'
  with C1 obtain Q' where C2: Q ↦ Cal* Q' and C3: (Q', R') ∈ Rel+
    by blast
  assume  $\bigwedge Q'. Q \mapsto \text{Cal}^* Q' \implies \exists P'. P \mapsto \text{Cal}^* P' \wedge (P', Q') \in \text{Rel}^+$ 
  with C2 obtain P' where C4: P ↦ Cal* P' and C5: (P', Q') ∈ Rel+
    by blast
  from C5 C3 have (P', R') ∈ Rel+
    by simp
  with C4 show  $\exists P'. P \mapsto \text{Cal}^* P' \wedge (P', R') \in \text{Rel}^+$ 
    by blast
qed
qed
from bisim C[where Rel=Rel] show weak-reduction-bisimulation (Rel+) Cal
  by blast
from A B[where Rel=Rel=] show weak-reduction-bisimulation (symcl (Rel=)) Cal
  by blast
from A C[where Rel=Rel=] show weak-reduction-bisimulation (Rel*) Cal
  using trancl-reflcl[of Rel]
  by auto
from A B[where Rel=Rel=] C[where Rel=symcl (Rel=)]
show weak-reduction-bisimulation ((symcl (Rel=))+) Cal
  by blast
qed

lemma weak-barbed-bisimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: weak-barbed-bisimulation Rel CWB
  shows weak-barbed-bisimulation (Rel=) CWB
    and weak-barbed-bisimulation (symcl Rel) CWB
    and weak-barbed-bisimulation (Rel+) CWB
    and weak-barbed-bisimulation (symcl (Rel=)) CWB
    and weak-barbed-bisimulation (Rel*) CWB
    and weak-barbed-bisimulation ((symcl (Rel=))+) CWB
proof -
  from bisim show weak-barbed-bisimulation (Rel=) CWB
    using weak-reduction-bisimulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
    weak-respection-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
    by fast
next
  from bisim show weak-barbed-bisimulation (symcl Rel) CWB
    using weak-reduction-bisimulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
    weak-respection-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
    by blast
next
  from bisim show weak-barbed-bisimulation (Rel+) CWB
    using weak-reduction-bisimulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
    weak-respection-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
    by blast
next
  from bisim show weak-barbed-bisimulation (symcl (Rel=)) CWB
    using weak-reduction-bisimulation-and-closures(4)[where Rel=Rel and Cal=Calculus CWB]
    weak-respection-of-barbs-and-closures(4)[where Rel=Rel and CWB=CWB]
    by blast
next
  from bisim show weak-barbed-bisimulation (Rel*) CWB
    using weak-reduction-bisimulation-and-closures(5)[where Rel=Rel and Cal=Calculus CWB]

```

```

      weak-respection-of-barbs-and-closures(5)[where Rel=Rel and CWB=CWB]
    by blast
next
from bisim show weak-barbed-bisimulation ((symcl (Rel=))+) CWB
  using weak-reduction-bisimulation-and-closures(6)[where Rel=Rel and Cal=Calculus CWB]
      weak-respection-of-barbs-and-closures(6)[where Rel=Rel and CWB=CWB]
  by blast
qed

```

A strong reduction bisimulation is relation  $R$  such that (1) if  $(P, Q)$  in  $R$  and  $P'$  is a derivative of  $P$  then there exists some  $Q'$  such that  $Q'$  is a derivative of  $Q$  and  $(P', Q')$  in  $R$ , and (2) if  $(P, Q)$  in  $R$  and  $Q'$  is a derivative of  $Q$  then there exists some  $P'$  such that  $P'$  is a derivative of  $P$  and  $(P', Q')$  in  $R$ .

**abbreviation** *strong-reduction-bisimulation*  
 $:: ('proc \times 'proc) set \Rightarrow 'proc \text{ processCalculus} \Rightarrow bool$   
**where**  
*strong-reduction-bisimulation*  $Rel \text{ } Cal \equiv$   
 $(\forall P \ Q \ P'. (P, Q) \in Rel \wedge P \mapsto Cal \ P' \longrightarrow (\exists Q'. Q \mapsto Cal \ Q' \wedge (P', Q') \in Rel))$   
 $\wedge (\forall P \ Q \ Q'. (P, Q) \in Rel \wedge Q \mapsto Cal \ Q' \longrightarrow (\exists P'. P \mapsto Cal \ P' \wedge (P', Q') \in Rel))$

A strong barbed bisimulation is strong reduction bisimulation that respects barbs.

**abbreviation** *strong-barbed-bisimulation*  
 $:: ('proc \times 'proc) set \Rightarrow ('proc, 'barbs) \text{ calculusWithBarbs} \Rightarrow bool$   
**where**  
*strong-barbed-bisimulation*  $Rel \text{ } CWB \equiv$   
*strong-reduction-bisimulation*  $Rel \text{ } (Calculus \ CWB) \wedge \text{rel-respects-barbs } Rel \text{ } CWB$

A symmetric strong simulation is a strong bisimulation.

**lemma** *symm-strong-reduction-simulation-is-bisimulation:*  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $Cal :: 'proc \text{ processCalculus}$   
**assumes**  $sym \ Rel$   
**and** *strong-reduction-simulation*  $Rel \text{ } Cal$   
**shows** *strong-reduction-bisimulation*  $Rel \text{ } Cal$   
**using**  $assms \ symD[of \ Rel]$   
**by** *blast*

**lemma** *symm-strong-barbed-simulation-is-bisimulation:*  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$   
**assumes**  $sym \ Rel$   
**and** *strong-barbed-simulation*  $Rel \text{ } CWB$   
**shows** *strong-barbed-bisimulation*  $Rel \text{ } CWB$   
**using**  $assms \ symD[of \ Rel]$   
**by** *blast*

If a relation as well as its inverse are strong simulations, then this relation is a strong bisimulation.

**lemma** *strong-reduction-simulations-impl-bisimulation:*  
**fixes**  $Rel :: ('proc \times 'proc) set$   
**and**  $Cal :: 'proc \text{ processCalculus}$   
**assumes**  $sim: \text{strong-reduction-simulation } Rel \text{ } Cal$   
**and**  $simInv: \text{strong-reduction-simulation } (Rel^{-1}) \text{ } Cal$   
**shows** *strong-reduction-bisimulation*  $Rel \text{ } Cal$   
**proof** *auto*  
**fix**  $P \ Q \ P'$   
**assume**  $(P, Q) \in Rel$  **and**  $P \mapsto Cal \ P'$   
**with**  $sim$  **show**  $\exists Q'. Q \mapsto Cal \ Q' \wedge (P', Q') \in Rel$   
**by** *simp*  
**next**

```

fix P Q Q'
assume (P, Q) ∈ Rel
hence (Q, P) ∈ Rel-1
  by simp
moreover assume Q ⟶Cal Q'
ultimately obtain P' where A1: P ⟶Cal P' and A2: (Q', P') ∈ Rel-1
  using simInv
  by blast
from A2 have (P', Q') ∈ Rel
  by induct
with A1 show ∃ P'. P ⟶Cal P' ∧ (P', Q') ∈ Rel
  by blast
qed

```

**lemma** *strong-reduction-bisimulations-impl-inverse-is-simulation:*

```

fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
assumes bisim: strong-reduction-bisimulation Rel Cal
shows strong-reduction-simulation (Rel-1) Cal
proof clarify
fix P Q P'
assume (Q, P) ∈ Rel
moreover assume P ⟶Cal P'
ultimately obtain Q' where A1: Q ⟶Cal Q' and A2: (Q', P') ∈ Rel
  using bisim
  by blast
from A2 have (P', Q') ∈ Rel-1
  by simp
with A1 show ∃ Q'. Q ⟶Cal Q' ∧ (P', Q') ∈ Rel-1
  by blast
qed

```

**lemma** *strong-reduction-simulations-iff-bisimulation:*

```

fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
shows (strong-reduction-simulation Rel Cal ∧ strong-reduction-simulation (Rel-1) Cal)
  = strong-reduction-bisimulation Rel Cal
  using strong-reduction-simulations-impl-bisimulation[where Rel=Rel and Cal=Cal]
  strong-reduction-bisimulations-impl-inverse-is-simulation[where Rel=Rel]
  by blast

```

**lemma** *strong-barbed-simulations-iff-bisimulation:*

```

fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
shows (strong-barbed-simulation Rel CWB ∧ strong-barbed-simulation (Rel-1) CWB)
  = strong-barbed-bisimulation Rel CWB
proof (rule iffI, erule conjE)
assume sim: strong-barbed-simulation Rel CWB
  and rev: strong-barbed-simulation (Rel-1) CWB
hence strong-reduction-bisimulation Rel (Calculus CWB)
  using strong-reduction-simulations-impl-bisimulation[where Rel=Rel and Cal=Calculus CWB]
  by blast
moreover from sim have rel-preserves-barbs Rel CWB
  by simp
moreover from rev have rel-reflects-barbs Rel CWB
  by simp
ultimately show strong-barbed-bisimulation Rel CWB
  by blast
next
assume bisim: strong-barbed-bisimulation Rel CWB
hence strong-barbed-simulation Rel CWB

```

by *blast*  
**moreover from** *bisim have strong-reduction-simulation*  $(Rel^{-1})$  (*Calculus CWB*)  
 using *strong-reduction-bisimulations-impl-inverse-is-simulation*[**where**  $Rel=Rel$ ]  
 by *simp*  
**moreover from** *bisim have rel-reflects-barbs*  $Rel$  *CWB*  
 by *blast*  
**hence** *rel-preserves-barbs*  $(Rel^{-1})$  *CWB*  
 by *simp*  
**ultimately**  
**show** *strong-barbed-simulation*  $Rel$  *CWB*  $\wedge$  *strong-barbed-simulation*  $(Rel^{-1})$  *CWB*  
 by *blast*  
**qed**

A strong bisimulation is a weak bisimulation.

**lemma** *strong-impl-weak-reduction-bisimulation*:

fixes  $Rel :: ('proc \times 'proc) \text{ set}$   
 and  $Cal :: 'proc \text{ processCalculus}$   
 assumes *bisim: strong-reduction-bisimulation*  $Rel$   $Cal$   
 shows *weak-reduction-bisimulation*  $Rel$   $Cal$

**proof**

**from** *bisim show weak-reduction-simulation*  $Rel$   $Cal$   
 using *strong-impl-weak-reduction-simulation*[**where**  $Rel=Rel$  **and**  $Cal=Cal$ ]  
 by *blast*

**next**

**show**  $\forall P Q Q'. (P, Q) \in Rel \wedge Q \mapsto_{Cal^*} Q' \longrightarrow (\exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q') \in Rel)$

**proof** *clarify*

**fix**  $P Q Q'$

**assume**  $A1: (P, Q) \in Rel$

**assume**  $Q \mapsto_{Cal^*} Q'$

**from** *this obtain n where*  $Q \mapsto_{Cal^n} Q'$

by *(auto simp add: steps-def)*

**thus**  $\exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q') \in Rel$

**proof** (*induct n arbitrary: Q'*)

**case**  $0$

**assume**  $Q \mapsto_{Cal^0} Q'$

**hence**  $Q = Q'$

by *(simp add: steps-refl)*

**moreover have**  $P \mapsto_{Cal^*} P$

by *(rule steps-refl)*

**ultimately show**  $\exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q') \in Rel$

using  $A1$

by *blast*

**next**

**case**  $(Suc\ n\ Q'')$

**assume**  $Q \mapsto_{Cal^{Suc\ n}} Q''$

**from** *this obtain Q' where*  $A2: Q \mapsto_{Cal^n} Q'$  **and**  $A3: Q' \mapsto_{Cal} Q''$

by *auto*

**assume**  $\bigwedge Q'. Q \mapsto_{Cal^n} Q' \implies \exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q') \in Rel$

**with**  $A2$  **obtain**  $P'$  **where**  $A4: P \mapsto_{Cal^*} P'$  **and**  $A5: (P', Q') \in Rel$

by *blast*

**from** *bisim*  $A5$   $A3$  **obtain**  $P''$  **where**  $A6: P' \mapsto_{Cal} P''$  **and**  $A7: (P'', Q'') \in Rel$

by *blast*

**from**  $A4$   $A6$  **have**  $P \mapsto_{Cal^*} P''$

using *steps-add*[**where**  $P=P$  **and**  $Q=P'$  **and**  $R=P''$ ]

by *(simp add: step-to-steps)*

**with**  $A7$  **show**  $\exists P'. P \mapsto_{Cal^*} P' \wedge (P', Q'') \in Rel$

by *blast*

**qed**

**qed**

**qed**

```

lemma strong-barbed-bisimulation-impl-weak-respection-of-barbs:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: strong-barbed-bisimulation Rel CWB
  shows rel-weakly-respects-barbs Rel CWB
proof
  from bisim show rel-weakly-preserves-barbs Rel CWB
    using strong-barbed-simulation-impl-weak-preservation-of-barbs[where Rel=Rel and CWB=CWB]
    by blast
next
  show rel-weakly-reflects-barbs Rel CWB
  proof clarify
    fix P Q a Q'
    assume  $(P, Q) \in \text{Rel}$  and  $Q \mapsto (\text{Calculus } CWB)^* Q'$ 
    with bisim obtain P' where  $A1: P \mapsto (\text{Calculus } CWB)^* P'$  and  $A2: (P', Q') \in \text{Rel}$ 
      using strong-impl-weak-reduction-bisimulation[where Rel=Rel and Cal=Calculus CWB]
      by blast
    assume  $Q' \Downarrow \langle CWB \rangle a$ 
    with bisim  $A2$  have  $P' \Downarrow \langle CWB \rangle a$ 
      by blast
    with  $A1$  show  $P \Downarrow \langle CWB \rangle a$ 
      by blast
  qed
qed

```

```

lemma strong-impl-weak-barbed-bisimulation:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: strong-barbed-bisimulation Rel CWB
  shows weak-barbed-bisimulation Rel CWB
    using bisim
      strong-impl-weak-reduction-bisimulation[where Rel=Rel and Cal=Calculus CWB]
      strong-barbed-bisimulation-impl-weak-respection-of-barbs[where Rel=Rel and CWB=CWB]
    by blast

```

The reflexive, symmetric, and/or transitive closure of a strong bisimulation is a strong bisimulation.

```

lemma strong-reduction-bisimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
    and Cal :: 'proc processCalculus
  assumes bisim: strong-reduction-bisimulation Rel Cal
  shows strong-reduction-bisimulation (Rel=) Cal
    and strong-reduction-bisimulation (symcl Rel) Cal
    and strong-reduction-bisimulation (Rel+) Cal
    and strong-reduction-bisimulation (symcl (Rel=)) Cal
    and strong-reduction-bisimulation (Rel*) Cal
    and strong-reduction-bisimulation ((symcl (Rel=))+) Cal
proof –
  from bisim show  $A: \text{strong-reduction-bisimulation } (Rel^=) \text{ Cal}$ 
    by (auto simp add: refl, blast+)
  have  $B: \bigwedge \text{Rel. strong-reduction-bisimulation Rel Cal}$ 
     $\implies \text{strong-reduction-bisimulation (symcl Rel) Cal}$ 
    by (auto simp add: symcl-def, blast+)
  from bisim  $B$ [where Rel=Rel] show strong-reduction-bisimulation (symcl Rel) Cal
    by blast
  have  $C: \bigwedge \text{Rel. strong-reduction-bisimulation Rel Cal}$ 
     $\implies \text{strong-reduction-bisimulation (Rel}^+\text{) Cal}$ 
proof
  fix Rel
  assume strong-reduction-bisimulation Rel Cal
  thus strong-reduction-simulation (Rel+) Cal

```

```

    using strong-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Cal]
  by blast
next
fix Rel
assume C1: strong-reduction-bisimulation Rel Cal
show  $\forall P Q Q'. (P, Q) \in Rel^+ \wedge Q \mapsto Cal Q' \rightarrow (\exists P'. P \mapsto Cal P' \wedge (P', Q') \in Rel^+)$ 
proof clarify
  fix P Q Q'
  assume (P, Q)  $\in Rel^+$  and  $Q \mapsto Cal Q'$ 
  thus  $\exists P'. P \mapsto Cal P' \wedge (P', Q') \in Rel^+$ 
  proof (induct arbitrary: Q')
    fix Q Q'
    assume (P, Q)  $\in Rel$  and  $Q \mapsto Cal Q'$ 
    with C1 obtain P' where  $P \mapsto Cal P'$  and  $(P', Q') \in Rel$ 
    by blast
    thus  $\exists P'. P \mapsto Cal P' \wedge (P', Q') \in Rel^+$ 
    by auto
  next
  case (step Q R R')
  assume (Q, R)  $\in Rel$  and  $R \mapsto Cal R'$ 
  with C1 obtain Q' where  $C2: Q \mapsto Cal Q'$  and  $C3: (Q', R') \in Rel^+$ 
  by blast
  assume  $\bigwedge Q'. Q \mapsto Cal Q' \implies \exists P'. P \mapsto Cal P' \wedge (P', Q') \in Rel^+$ 
  with C2 obtain P' where  $C4: P \mapsto Cal P'$  and  $C5: (P', Q') \in Rel^+$ 
  by blast
  from C5 C3 have  $(P', R') \in Rel^+$ 
  by simp
  with C4 show  $\exists P'. P \mapsto Cal P' \wedge (P', R') \in Rel^+$ 
  by blast
qed
qed
qed
from bisim C[where Rel=Rel] show strong-reduction-bisimulation  $(Rel^+)$  Cal
  by blast
from A B[where Rel=Rel=]
show strong-reduction-bisimulation (symcl  $(Rel^=)$ ) Cal
  by blast
from A C[where Rel=Rel=]
show strong-reduction-bisimulation  $(Rel^*)$  Cal
  using trancl-reflcl[of Rel]
  by auto
from A B[where Rel=Rel=] C[where Rel=symcl  $(Rel^=)$ ]
show strong-reduction-bisimulation  $((symcl (Rel^=))^+)$  Cal
  by blast
qed

lemma strong-barbed-bisimulation-and-closures:
  fixes Rel :: ('proc  $\times$  'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: strong-barbed-bisimulation Rel CWB
  shows strong-barbed-bisimulation  $(Rel^=)$  CWB
  and strong-barbed-bisimulation (symcl Rel) CWB
  and strong-barbed-bisimulation  $(Rel^+)$  CWB
  and strong-barbed-bisimulation (symcl  $(Rel^=)$ ) CWB
  and strong-barbed-bisimulation  $(Rel^*)$  CWB
  and strong-barbed-bisimulation  $((symcl (Rel^=))^+)$  CWB
proof -
  from bisim show strong-barbed-bisimulation  $(Rel^=)$  CWB
  using strong-reduction-bisimulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
  respecton-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]

```

```

  by fast
next
  from bisim show strong-barbed-bisimulation (symcl Rel) CWB
    using strong-reduction-bisimulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
      respect-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
  by blast
next
  from bisim show strong-barbed-bisimulation (Rel+) CWB
    using strong-reduction-bisimulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
      respect-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
  by blast
next
  from bisim show strong-barbed-bisimulation (symcl (Rel=)) CWB
    using strong-reduction-bisimulation-and-closures(4)[where Rel=Rel and Cal=Calculus CWB]
      respect-of-barbs-and-closures(4)[where Rel=Rel and CWB=CWB]
  by blast
next
  from bisim show strong-barbed-bisimulation (Rel*) CWB
    using strong-reduction-bisimulation-and-closures(5)[where Rel=Rel and Cal=Calculus CWB]
      respect-of-barbs-and-closures(5)[where Rel=Rel and CWB=CWB]
  by blast
next
  from bisim show strong-barbed-bisimulation ((symcl (Rel=))+) CWB
    using strong-reduction-bisimulation-and-closures(6)[where Rel=Rel and Cal=Calculus CWB]
      respect-of-barbs-and-closures(6)[where Rel=Rel and CWB=CWB]
  by blast
qed

```

### 3.6 Step Closure of Relations

The step closure of a relation on process terms is the transitive closure of the union of the relation and the inverse of the reduction relation of the respective calculus.

```

inductive-set stepsClosure :: ('a × 'a) set ⇒ 'a processCalculus ⇒ ('a × 'a) set
  for Rel :: ('a × 'a) set
  and Cal :: 'a processCalculus
  where
  rel: (P, Q) ∈ Rel ⇒ (P, Q) ∈ stepsClosure Rel Cal |
  steps: P ↦ Cal* P' ⇒ (P', P) ∈ stepsClosure Rel Cal |
  trans: [(P, Q) ∈ stepsClosure Rel Cal; (Q, R) ∈ stepsClosure Rel Cal]
    ⇒ (P, R) ∈ stepsClosure Rel Cal

```

```

abbreviation stepsClosureInfix ::
  'a ⇒ ('a × 'a) set ⇒ 'a processCalculus ⇒ 'a ⇒ bool (- R↦<-,-> - [75, 75, 75, 75] 80)
  where
  P R↦<Rel,Cal> Q ≡ (P, Q) ∈ stepsClosure Rel Cal

```

Applying the steps closure twice does not change the relation.

```

lemma steps-closure-of-steps-closure:
  fixes Rel :: ('a × 'a) set
  and Cal :: 'a processCalculus
  shows stepsClosure (stepsClosure Rel Cal) Cal = stepsClosure Rel Cal
proof auto
  fix P Q
  assume P R↦<stepsClosure Rel Cal,Cal> Q
  thus P R↦<Rel,Cal> Q
  proof induct
  case (rel P Q)
  assume P R↦<Rel,Cal> Q
  thus P R↦<Rel,Cal> Q
  by simp

```



```

next
  case (steps P P')
  assume P  $\mapsto$  Cal* P'
  thus P'  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  P
    by (rule stepsClosure.steps)
next
  case (trans P Q R)
  assume P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  Q and Q  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  R
  thus P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  R
    by (rule stepsClosure.trans)
qed
next
fix P Q
assume P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  Q
thus P  $\mathcal{R} \mapsto \langle \text{stepsClosure Rel Cal}, \text{Cal} \rangle$  Q
  by (rule stepsClosure.rel)
qed

```

The steps closure is a preorder.

```

lemma stepsClosure-refl:
  fixes Rel :: ('a  $\times$  'a) set
  and Cal :: 'a processCalculus
  shows refl (stepsClosure Rel Cal)
    unfolding refl-on-def
proof auto
fix P
have P  $\mapsto$  Cal* P
  by (rule steps-refl)
thus P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  P
  by (rule stepsClosure.steps)
qed

```

```

lemma refl-trans-closure-of-rel-impl-steps-closure:
  fixes Rel :: ('a  $\times$  'a) set
  and Cal :: 'a processCalculus
  and P Q :: 'a
  assumes (P, Q)  $\in$  Rel*
  shows P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  Q
    using assms
proof induct
show P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  P
  using stepsClosure-refl[of Rel Cal]
  unfolding refl-on-def
  by simp
next
case (step Q R)
assume (Q, R)  $\in$  Rel and P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  Q
thus P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  R
  using stepsClosure.rel[of Q R Rel Cal] stepsClosure.trans[of P Q Rel Cal R]
  by blast
qed

```

The steps closure of a relation is always a weak reduction simulation.

```

lemma steps-closure-is-weak-reduction-simulation:
  fixes Rel :: ('a  $\times$  'a) set
  and Cal :: 'a processCalculus
  shows weak-reduction-simulation (stepsClosure Rel Cal) Cal
proof clarify
fix P Q P'
assume P  $\mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle$  Q and P  $\mapsto$  Cal* P'

```

**thus**  $\exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q'$   
**proof** (*induct arbitrary: P'*)  
**case** (*rel P Q*)  
**assume**  $P \mapsto_{\text{Cal}^*} P'$   
**hence**  $P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} P$   
**by** (*rule stepsClosure.steps*)  
**moreover assume**  $(P, Q) \in \text{Rel}$   
**hence**  $P \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q$   
**by** (*simp add: stepsClosure.rel*)  
**ultimately have**  $P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q$   
**by** (*rule stepsClosure.trans*)  
**thus**  $\exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q'$   
**using** *steps-refl*[**where**  $\text{Cal} = \text{Cal}$  **and**  $P = Q$ ]  
**by** *blast*  
**next**  
**case** (*steps P P' P''*)  
**assume**  $P \mapsto_{\text{Cal}^*} P'$  **and**  $P' \mapsto_{\text{Cal}^*} P''$   
**hence**  $P \mapsto_{\text{Cal}^*} P''$   
**by** (*rule steps-add*)  
**moreover have**  $P'' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} P''$   
**using** *stepsClosure-refl*[**where**  $\text{Rel} = \text{Rel}$  **and**  $\text{Cal} = \text{Cal}$ ]  
**unfolding** *refl-on-def*  
**by** *simp*  
**ultimately show**  $\exists Q'. P \mapsto_{\text{Cal}^*} Q' \wedge P'' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q'$   
**by** *blast*  
**next**  
**case** (*trans P Q R*)  
**assume**  $P \mapsto_{\text{Cal}^*} P'$   
**and**  $\bigwedge P'. P \mapsto_{\text{Cal}^*} P' \implies \exists Q'. Q \mapsto_{\text{Cal}^*} Q' \wedge P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q'$   
**from this obtain**  $Q'$  **where**  $A1: Q \mapsto_{\text{Cal}^*} Q'$  **and**  $A2: P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} Q'$   
**by** *blast*  
**assume**  $\bigwedge Q'. Q \mapsto_{\text{Cal}^*} Q' \implies \exists R'. R \mapsto_{\text{Cal}^*} R' \wedge Q' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} R'$   
**with A1 obtain**  $R'$  **where**  $A3: R \mapsto_{\text{Cal}^*} R'$  **and**  $A4: Q' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} R'$   
**by** *blast*  
**from A2 A4 have**  $P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} R'$   
**by** (*rule stepsClosure.trans*)  
**with A3 show**  $\exists R'. R \mapsto_{\text{Cal}^*} R' \wedge P' \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} R'$   
**by** *blast*  
**qed**  
**qed**

If  $\text{Rel}$  is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of  $\text{Rel}$  is a contrasimulation.

**lemma** *inverse-contrasimulation-impl-reverse-pair-in-steps-closure*:

**fixes**  $\text{Rel} :: ('a \times 'a) \text{ set}$   
**and**  $\text{Cal} :: 'a \text{ processCalculus}$   
**and**  $P Q :: 'a$   
**assumes** *con*: *weak-reduction-contrasimulation*  $(\text{Rel}^{-1}) \text{ Cal}$   
**and** *pair*:  $(P, Q) \in \text{Rel}$   
**shows**  $Q \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} P$   
**proof** –  
**from** *pair* **have**  $(Q, P) \in \text{Rel}^{-1}$   
**by** *simp*  
**moreover have**  $Q \mapsto_{\text{Cal}^*} Q$   
**by** (*rule steps-refl*)  
**ultimately obtain**  $P'$  **where**  $A1: P \mapsto_{\text{Cal}^*} P'$  **and**  $A2: (P', Q) \in \text{Rel}^{-1}$   
**using** *con*  
**by** *blast*  
**from A2 have**  $Q \mathcal{R} \mapsto_{\langle \text{Rel}, \text{Cal} \rangle} P'$   
**by** (*simp add: stepsClosure.rel*)

**moreover from  $A1$  have  $P' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P$**   
 by (rule *stepsClosure.steps*)  
**ultimately show  $Q \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P$**   
 by (rule *stepsClosure.trans*)  
**qed**

**lemma *simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation*:**

**fixes  $\text{Rel} :: ('a \times 'a)$  set**  
**and  $\text{Cal} :: 'a$  processCalculus**  
**assumes  $\text{sim}$ : weak-reduction-simulation  $\text{Rel} \text{Cal}$**   
**and  $\text{con}$ : weak-reduction-contrasimulation  $(\text{Rel}^{-1}) \text{Cal}$**   
**shows weak-reduction-contrasimulation  $(\text{stepsClosure} \text{Rel} \text{Cal}) \text{Cal}$**   
**proof clarify**  
**fix  $P Q P'$**   
**assume  $P \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle Q$  and  $P \mapsto \text{Cal}^* P'$**   
**thus  $\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**proof (induct arbitrary:  $P'$ )**  
**case (rel  $P Q$ )**  
**assume  $(P, Q) \in \text{Rel}$  and  $P \mapsto \text{Cal}^* P'$**   
**with  $\text{sim}$  obtain  $Q'$  where  $A1$ :  $Q \mapsto \text{Cal}^* Q'$  and  $A2$ :  $(P', Q') \in \text{Rel}$**   
**by blast**  
**from  $A2$  con have  $Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**using *inverse-contrasimulation-impl-reverse-pair-in-steps-closure* [where  $\text{Rel} = \text{Rel}$ ]**  
**by blast**  
**with  $A1$  show  $\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**by blast**  
**next**  
**case (steps  $P P' P''$ )**  
**assume  $P \mapsto \text{Cal}^* P'$  and  $P' \mapsto \text{Cal}^* P''$**   
**hence  $P \mapsto \text{Cal}^* P''$**   
**by (rule *steps-add*)**  
**thus  $\exists Q'. P \mapsto \text{Cal}^* Q' \wedge Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P''$**   
**using *stepsClosure-refl* [where  $\text{Rel} = \text{Rel}$  and  $\text{Cal} = \text{Cal}$ ]**  
**unfolding *refl-on-def***  
**by blast**  
**next**  
**case (trans  $P Q R$ )**  
**assume  $\bigwedge P'. P \mapsto \text{Cal}^* P' \implies \exists Q'. Q \mapsto \text{Cal}^* Q' \wedge Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**and  $P \mapsto \text{Cal}^* P'$**   
**from this obtain  $Q'$  where  $A1$ :  $Q \mapsto \text{Cal}^* Q'$  and  $A2$ :  $Q' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**by blast**  
**assume  $\bigwedge Q'. Q \mapsto \text{Cal}^* Q' \implies \exists R'. R \mapsto \text{Cal}^* R' \wedge R' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle Q'$**   
**with  $A1$  obtain  $R'$  where  $A3$ :  $R \mapsto \text{Cal}^* R'$  and  $A4$ :  $R' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle Q'$**   
**by blast**  
**from  $A4$   $A2$  have  $R' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**by (rule *stepsClosure.trans*)**  
**with  $A3$  show  $\exists R'. R \mapsto \text{Cal}^* R' \wedge R' \mathcal{R} \mapsto \langle \text{Rel}, \text{Cal} \rangle P'$**   
**by blast**  
**qed**  
**qed**

Accordingly, if  $\text{Rel}$  is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of  $\text{Rel}$  is a coupled simulation.

**lemma *simulation-and-inverse-contrasimulation-impl-steps-closure-is-coupled-simulation*:**

**fixes  $\text{Rel} :: ('a \times 'a)$  set**  
**and  $\text{Cal} :: 'a$  processCalculus**  
**assumes  $\text{sim}$ : weak-reduction-simulation  $\text{Rel} \text{Cal}$**   
**and  $\text{con}$ : weak-reduction-contrasimulation  $(\text{Rel}^{-1}) \text{Cal}$**   
**shows weak-reduction-coupled-simulation  $(\text{stepsClosure} \text{Rel} \text{Cal}) \text{Cal}$**   
**using  $\text{sim}$  con *simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation***

*steps-closure-is-weak-reduction-simulation*[**where**  $Rel=Rel$  **and**  $Cal=Cal$ ]  
**by** *simp*

If the relation that is closed under steps is a (contra)simulation, then we can conclude from a pair in the closure on a pair in the original relation.

**lemma** *stepsClosure-simulation-impl-refl-trans-closure-of-Rel*:

**fixes**  $Rel :: ('a \times 'a)$  *set*

**and**  $Cal :: 'a$  *processCalculus*

**and**  $P\ Q :: 'a$

**assumes**  $A1: P \mathcal{R} \mapsto \langle Rel, Cal \rangle Q$

**and**  $A2: \text{weak-reduction-simulation } Rel\ Cal$

**shows**  $\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P, Q') \in Rel^*$

**proof** –

**have**  $\forall P'. P \mapsto_{Cal^*} P' \longrightarrow (\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^*)$

**using**  $A1$

**proof** *induct*

**case** (*rel*  $P\ Q$ )

**assume**  $(P, Q) \in Rel$

**with**  $A2$  **have**  $\forall P'. P \mapsto_{Cal^*} P' \longrightarrow (\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel)$

**by** *blast*

**thus**  $\forall P'. P \mapsto_{Cal^*} P' \longrightarrow (\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^*)$

**by** *blast*

**next**

**case** (*steps*  $P\ P'$ )

**assume**  $A: P \mapsto_{Cal^*} P'$

**show**  $\forall P''. P' \mapsto_{Cal^*} P'' \longrightarrow (\exists Q'. P \mapsto_{Cal^*} Q' \wedge (P'', Q') \in Rel^*)$

**proof** *clarify*

**fix**  $P''$

**assume**  $P' \mapsto_{Cal^*} P''$

**with**  $A$  **have**  $P \mapsto_{Cal^*} P''$

**by** (*rule steps-add*)

**moreover** **have**  $(P'', P'') \in Rel^*$

**by** *simp*

**ultimately** **show**  $\exists Q'. P \mapsto_{Cal^*} Q' \wedge (P'', Q') \in Rel^*$

**by** *blast*

**qed**

**next**

**case** (*trans*  $P\ Q\ R$ )

**assume**  $A1: \forall P'. P \mapsto_{Cal^*} P' \longrightarrow (\exists Q'. Q \mapsto_{Cal^*} Q' \wedge (P', Q') \in Rel^*)$

**and**  $A2: \forall Q'. Q \mapsto_{Cal^*} Q' \longrightarrow (\exists R'. R \mapsto_{Cal^*} R' \wedge (Q', R') \in Rel^*)$

**show**  $\forall P'. P \mapsto_{Cal^*} P' \longrightarrow (\exists R'. R \mapsto_{Cal^*} R' \wedge (P', R') \in Rel^*)$

**proof** *clarify*

**fix**  $P'$

**assume**  $P \mapsto_{Cal^*} P'$

**with**  $A1$  **obtain**  $Q'$  **where**  $A3: Q \mapsto_{Cal^*} Q'$  **and**  $A4: (P', Q') \in Rel^*$

**by** *blast*

**from**  $A2\ A3$  **obtain**  $R'$  **where**  $A5: R \mapsto_{Cal^*} R'$  **and**  $A6: (Q', R') \in Rel^*$

**by** *blast*

**from**  $A4\ A6$  **have**  $(P', R') \in Rel^*$

**by** *simp*

**with**  $A5$  **show**  $\exists R'. R \mapsto_{Cal^*} R' \wedge (P', R') \in Rel^*$

**by** *blast*

**qed**

**qed**

**moreover** **have**  $P \mapsto_{Cal^*} P$

**by** (*rule steps-refl*)

**ultimately** **show** *?thesis*

**by** *blast*

**qed**

**lemma** *stepsClosure-contrasimulation-impl-refl-trans-closure-of-Rel*:

**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**and**  $Cal :: 'a \text{ processCalculus}$   
**and**  $P Q :: 'a$   
**assumes**  $A1: P \mathcal{R} \mapsto \langle Rel, Cal \rangle Q$   
**and**  $A2: \text{weak-reduction-contrasimulation } Rel \text{ } Cal$   
**shows**  $\exists Q'. Q \mapsto Cal^* Q' \wedge (Q', P) \in Rel^*$

**proof** –

**have**  $\forall P'. P \mapsto Cal^* P' \longrightarrow (\exists Q'. Q \mapsto Cal^* Q' \wedge (Q', P') \in Rel^*)$   
**using**  $A1$

**proof** *induct*

**case**  $(rel \ P \ Q)$   
**assume**  $(P, Q) \in Rel$   
**with**  $A2$  **have**  $\forall P'. P \mapsto Cal^* P' \longrightarrow (\exists Q'. Q \mapsto Cal^* Q' \wedge (Q', P') \in Rel)$   
**by** *blast*

**thus**  $\forall P'. P \mapsto Cal^* P' \longrightarrow (\exists Q'. Q \mapsto Cal^* Q' \wedge (Q', P') \in Rel^*)$   
**by** *blast*

**next**

**case**  $(steps \ P \ P')$   
**assume**  $A: P \mapsto Cal^* P'$   
**show**  $\forall P''. P' \mapsto Cal^* P'' \longrightarrow (\exists Q'. P \mapsto Cal^* Q' \wedge (Q', P'') \in Rel^*)$

**proof** *clarify*

**fix**  $P''$   
**assume**  $P' \mapsto Cal^* P''$   
**with**  $A$  **have**  $P \mapsto Cal^* P''$   
**by**  $(rule \ steps\text{-}add)$   
**moreover** **have**  $(P'', P') \in Rel^*$   
**by** *simp*  
**ultimately** **show**  $\exists Q'. P \mapsto Cal^* Q' \wedge (Q', P'') \in Rel^*$   
**by** *blast*

**qed**

**next**

**case**  $(trans \ P \ Q \ R)$   
**assume**  $A1: \forall P'. P \mapsto Cal^* P' \longrightarrow (\exists Q'. Q \mapsto Cal^* Q' \wedge (Q', P') \in Rel^*)$   
**and**  $A2: \forall Q'. Q \mapsto Cal^* Q' \longrightarrow (\exists R'. R \mapsto Cal^* R' \wedge (R', Q') \in Rel^*)$   
**show**  $\forall P'. P \mapsto Cal^* P' \longrightarrow (\exists R'. R \mapsto Cal^* R' \wedge (R', P') \in Rel^*)$

**proof** *clarify*

**fix**  $P'$   
**assume**  $P \mapsto Cal^* P'$   
**with**  $A1$  **obtain**  $Q'$  **where**  $A3: Q \mapsto Cal^* Q'$  **and**  $A4: (Q', P') \in Rel^*$   
**by** *blast*  
**from**  $A2 \ A3$  **obtain**  $R'$  **where**  $A5: R \mapsto Cal^* R'$  **and**  $A6: (R', Q') \in Rel^*$   
**by** *blast*  
**from**  $A4 \ A6$  **have**  $(R', P') \in Rel^*$   
**by** *simp*  
**with**  $A5$  **show**  $\exists R'. R \mapsto Cal^* R' \wedge (R', P') \in Rel^*$   
**by** *blast*

**qed**

**qed**

**moreover** **have**  $P \mapsto Cal^* P$   
**by**  $(rule \ steps\text{-}refl)$   
**ultimately** **show** *?thesis*  
**by** *blast*

**qed**

**lemma** *stepsClosure-contrasimulation-of-inverse-impl-refl-trans-closure-of-Rel*:

**fixes**  $Rel :: ('a \times 'a) \text{ set}$   
**and**  $Cal :: 'a \text{ processCalculus}$   
**and**  $P Q :: 'a$   
**assumes**  $A1: P \mathcal{R} \mapsto \langle Rel^{-1}, Cal \rangle Q$   
**and**  $A2: \text{weak-reduction-contrasimulation } (Rel^{-1}) \text{ } Cal$

```

shows  $\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (P, Q') \in \text{Rel}^*$ 
proof -
have  $\forall P'. P \mapsto \text{Cal}^* P' \longrightarrow (\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (P', Q') \in \text{Rel}^*)$ 
  using A1
proof induct
  case (rel P Q)
  assume  $(P, Q) \in \text{Rel}^{-1}$ 
  with A2 have  $\forall P'. P \mapsto \text{Cal}^* P' \longrightarrow (\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (Q', P') \in \text{Rel}^{-1})$ 
    by blast
  thus  $\forall P'. P \mapsto \text{Cal}^* P' \longrightarrow (\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (P', Q') \in \text{Rel}^*)$ 
    by blast
next
  case (steps P P')
  assume A:  $P \mapsto \text{Cal}^* P'$ 
  show  $\forall P''. P' \mapsto \text{Cal}^* P'' \longrightarrow (\exists Q'. P \mapsto \text{Cal}^* Q' \wedge (P'', Q') \in \text{Rel}^*)$ 
proof clarify
  fix P''
  assume  $P' \mapsto \text{Cal}^* P''$ 
  with A have  $P \mapsto \text{Cal}^* P''$ 
    by (rule steps-add)
  moreover have  $(P'', P') \in \text{Rel}^*$ 
    by simp
  ultimately show  $\exists Q'. P \mapsto \text{Cal}^* Q' \wedge (P'', Q') \in \text{Rel}^*$ 
    by blast
qed
next
  case (trans P Q R)
  assume A1:  $\forall P'. P \mapsto \text{Cal}^* P' \longrightarrow (\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (P', Q') \in \text{Rel}^*)$ 
    and A2:  $\forall Q'. Q \mapsto \text{Cal}^* Q' \longrightarrow (\exists R'. R \mapsto \text{Cal}^* R' \wedge (Q', R') \in \text{Rel}^*)$ 
  show  $\forall P'. P \mapsto \text{Cal}^* P' \longrightarrow (\exists R'. R \mapsto \text{Cal}^* R' \wedge (P', R') \in \text{Rel}^*)$ 
proof clarify
  fix P'
  assume  $P \mapsto \text{Cal}^* P'$ 
  with A1 obtain Q' where A3:  $Q \mapsto \text{Cal}^* Q'$  and A4:  $(P', Q') \in \text{Rel}^*$ 
    by blast
  from A3 A2 obtain R' where A5:  $R \mapsto \text{Cal}^* R'$  and A6:  $(Q', R') \in \text{Rel}^*$ 
    by blast
  from A4 A6 have  $(P', R') \in \text{Rel}^*$ 
    by simp
  with A5 show  $\exists R'. R \mapsto \text{Cal}^* R' \wedge (P', R') \in \text{Rel}^*$ 
    by blast
qed
qed
moreover have  $P \mapsto \text{Cal}^* P$ 
  by (rule steps-refl)
ultimately show ?thesis
  by blast
qed

end
theory Encodings
  imports ProcessCalculi
begin

```

## 4 Encodings

In the simplest case an encoding from a source into a target language is a mapping from source into target terms. Encodability criteria describe properties on such mappings. To analyse encodability criteria we map them on conditions on relations between source and target terms. More precisely, we

consider relations on pairs of the disjoint union of source and target terms. We denote this disjoint union of source and target terms by *Proc*.

**datatype** (*'procS*, *'procT*) *Proc* =  
*SourceTerm 'procS* |  
*TargetTerm 'procT*

**definition** *STCal*

:: *'procS processCalculus*  $\Rightarrow$  *'procT processCalculus*  
 $\Rightarrow$  ((*'procS*, *'procT*) *Proc*) *processCalculus*

**where**

*STCal Source Target*  $\equiv$

( $\Downarrow$ *Reductions* =  $\lambda P P'$ .)

( $\exists SP SP'. P = \text{SourceTerm } SP \wedge P' = \text{SourceTerm } SP' \wedge \text{Reductions Source } SP SP'$ )  $\vee$

( $\exists TP TP'. P = \text{TargetTerm } TP \wedge P' = \text{TargetTerm } TP' \wedge \text{Reductions Target } TP TP'$ )

**definition** *STCalWB*

:: (*'procS*, *'barbs*) *calculusWithBarbs*  $\Rightarrow$  (*'procT*, *'barbs*) *calculusWithBarbs*  
 $\Rightarrow$  ((*'procS*, *'procT*) *Proc*, *'barbs*) *calculusWithBarbs*

**where**

*STCalWB Source Target*  $\equiv$

(*Calculus* = *STCal (calculusWithBarbs.Calculus Source) (calculusWithBarbs.Calculus Target)*),

*HasBarb* =  $\lambda P a. (\exists SP. P = \text{SourceTerm } SP \wedge (\text{calculusWithBarbs.HasBarb Source } SP a) \vee$

( $\exists TP. P = \text{TargetTerm } TP \wedge (\text{calculusWithBarbs.HasBarb Target } TP a)$ )

An encoding consists of a source language, a target language, and a mapping from source into target terms.

**locale** *encoding* =

**fixes** *Source* :: *'procS processCalculus*

**and** *Target* :: *'procT processCalculus*

**and** *Enc* :: *'procS*  $\Rightarrow$  *'procT*

**begin**

**abbreviation** *enc* :: *'procS*  $\Rightarrow$  *'procT* ( $\llbracket - \rrbracket$  [65] 70) **where**

$\llbracket S \rrbracket \equiv \text{Enc } S$

**abbreviation** *isSource* :: (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* ( $- \in \text{ProcS}$  [70] 80) **where**

$P \in \text{ProcS} \equiv (\exists S. P = \text{SourceTerm } S)$

**abbreviation** *isTarget* :: (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* ( $- \in \text{ProcT}$  [70] 80) **where**

$P \in \text{ProcT} \equiv (\exists T. P = \text{TargetTerm } T)$

**abbreviation** *getSource*

:: *'procS*  $\Rightarrow$  (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* ( $- \in S$  - [70, 70] 80)

**where**

$S \in S P \equiv (P = \text{SourceTerm } S)$

**abbreviation** *getTarget*

:: *'procT*  $\Rightarrow$  (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* ( $- \in T$  - [70, 70] 80)

**where**

$T \in T P \equiv (P = \text{TargetTerm } T)$

A step of a term in *Proc* is either a source term step or a target term step.

**abbreviation** *stepST*

:: (*'procS*, *'procT*) *Proc*  $\Rightarrow$  (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* ( $- \mapsto ST$  - [70, 70] 80)

**where**

$P \mapsto ST P' \equiv$

( $\exists S S'. S \in S P \wedge S' \in S P' \wedge S \mapsto \text{Source } S'$ )  $\vee$  ( $\exists T T'. T \in T P \wedge T' \in T P' \wedge T \mapsto \text{Target } T'$ )

**lemma** *stepST-STCal-step*:

**fixes** *P P'* :: (*'procS*, *'procT*) *Proc*

**shows**  $P \mapsto (STCal\ Source\ Target)\ P' = P \mapsto ST\ P'$   
**by** (*simp add: STCal-def*)

**lemma** *STStep-step*:

**fixes**  $S :: 'procS$   
**and**  $T :: 'procT$   
**and**  $P' :: ('procS, 'procT)\ Proc$   
**shows**  $SourceTerm\ S \mapsto ST\ P' = (\exists S'. S' \in S\ P' \wedge S \mapsto Source\ S')$   
**and**  $TargetTerm\ T \mapsto ST\ P' = (\exists T'. T' \in T\ P' \wedge T \mapsto Target\ T')$   
**by** *blast+*

**lemma** *STCal-step*:

**fixes**  $S :: 'procS$   
**and**  $T :: 'procT$   
**and**  $P' :: ('procS, 'procT)\ Proc$   
**shows**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)\ P' = (\exists S'. S' \in S\ P' \wedge S \mapsto Source\ S')$   
**and**  $TargetTerm\ T \mapsto (STCal\ Source\ Target)\ P' = (\exists T'. T' \in T\ P' \wedge T \mapsto Target\ T')$   
**by** (*simp add: STCal-def*)**+**

A sequence of steps of a term in Proc is either a sequence of source term steps or a sequence of target term steps.

**abbreviation** *stepsST*

$:: ('procS, 'procT)\ Proc \Rightarrow ('procS, 'procT)\ Proc \Rightarrow bool\ (- \mapsto ST* - [70, 70]\ 80)$   
**where**  
 $P \mapsto ST* P' \equiv$   
 $(\exists S\ S'. S \in S\ P \wedge S' \in S\ P' \wedge S \mapsto Source* S') \vee (\exists T\ T'. T \in T\ P \wedge T' \in T\ P' \wedge T \mapsto Target* T')$

**lemma** *STSteps-steps*:

**fixes**  $S :: 'procS$   
**and**  $T :: 'procT$   
**and**  $P' :: ('procS, 'procT)\ Proc$   
**shows**  $SourceTerm\ S \mapsto ST* P' = (\exists S'. S' \in S\ P' \wedge S \mapsto Source* S')$   
**and**  $TargetTerm\ T \mapsto ST* P' = (\exists T'. T' \in T\ P' \wedge T \mapsto Target* T')$   
**by** *blast+*

**lemma** *STCal-steps*:

**fixes**  $S :: 'procS$   
**and**  $T :: 'procT$   
**and**  $P' :: ('procS, 'procT)\ Proc$   
**shows**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)* P' = (\exists S'. S' \in S\ P' \wedge S \mapsto Source* S')$   
**and**  $TargetTerm\ T \mapsto (STCal\ Source\ Target)* P' = (\exists T'. T' \in T\ P' \wedge T \mapsto Target* T')$

**proof** *auto*

**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)* P'$   
**from this obtain**  $n$  **where**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)^n P'$   
**by** (*auto simp add: steps-def*)  
**thus**  $\exists S'. S' \in S\ P' \wedge S \mapsto Source* S'$   
**proof** (*induct n arbitrary: P'*)  
**case**  $0$   
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)^0 P'$   
**hence**  $S \in S\ P'$   
**by** *simp*  
**moreover have**  $S \mapsto Source* S$   
**by** (*rule steps-refl*)  
**ultimately show**  $\exists S'. S' \in S\ P' \wedge S \mapsto Source* S'$   
**by** *blast*

**next**

**case** ( $Suc\ n\ P''$ )  
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)^{Suc\ n}\ P''$   
**from this obtain**  $P'$  **where**  $A1: SourceTerm\ S \mapsto (STCal\ Source\ Target)^n P'$   
**and**  $A2: P' \mapsto (STCal\ Source\ Target)\ P''$



by *auto*  
**assume**  $\bigwedge P'. \text{SourceTerm } S \mapsto (\text{STCal Source Target})^n P' \implies \exists S'. S' \in S P' \wedge S \mapsto \text{Source}^* S'$   
**with**  $A1$  **obtain**  $S'$  **where**  $A3: S' \in S P'$  **and**  $A4: S \mapsto \text{Source}^* S'$   
 by *blast*  
**from**  $A2 A3$  **obtain**  $S''$  **where**  $A5: S'' \in S P''$  **and**  $A6: S' \mapsto \text{Source } S''$   
 using  $\text{STCal-step}(1)[\text{where } S=S' \text{ and } P'=P'']$   
 by *blast*  
**from**  $A4 A6$  **have**  $S \mapsto \text{Source}^* S''$   
 using  $\text{step-to-steps}[\text{where } \text{Cal}=\text{Source} \text{ and } P=S' \text{ and } P'=S'']$   
 by (*simp add: steps-add*[**where**  $\text{Cal}=\text{Source}$  **and**  $P=S$  **and**  $Q=S'$  **and**  $R=S''$ ])  
**with**  $A5$  **show**  $\exists S''. S'' \in S P'' \wedge S \mapsto \text{Source}^* S''$   
 by *blast*  
**qed**  
**next**  
**fix**  $S'$   
**assume**  $S \mapsto \text{Source}^* S'$   
**from** *this* **obtain**  $n$  **where**  $S \mapsto \text{Source}^n S'$   
 by (*auto simp add: steps-def*)  
**thus**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
**proof** (*induct n arbitrary: S'*)  
 case 0  
**assume**  $S \mapsto \text{Source}^0 S'$   
**hence**  $S = S'$   
 by *auto*  
**thus**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
 by (*simp add: steps-refl*)  
**next**  
 case (*Suc n S''*)  
**assume**  $S \mapsto \text{Source}^{\text{Suc } n} S''$   
**from** *this* **obtain**  $S'$  **where**  $B1: S \mapsto \text{Source}^n S'$  **and**  $B2: S' \mapsto \text{Source } S''$   
 by *auto*  
**assume**  $\bigwedge S'. S \mapsto \text{Source}^n S' \implies \text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
**with**  $B1$  **have**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
 by *blast*  
**moreover** **from**  $B2$  **have**  $\text{SourceTerm } S' \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S'')$   
 using  $\text{step-to-steps}[\text{where } \text{Cal}=\text{STCal Source Target} \text{ and } P=\text{SourceTerm } S']$   
 by (*simp add: STCal-def*)  
**ultimately** **show**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S'')$   
 by (*rule steps-add*)  
**qed**  
**next**  
**assume**  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^* P'$   
**from** *this* **obtain**  $n$  **where**  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^n P'$   
 by (*auto simp add: steps-def*)  
**thus**  $\exists T'. T' \in T P' \wedge T \mapsto \text{Target}^* T'$   
**proof** (*induct n arbitrary: P'*)  
 case 0  
**assume**  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^0 P'$   
**hence**  $T \in T P'$   
 by *simp*  
**moreover** **have**  $T \mapsto \text{Target}^* T$   
 by (*rule steps-refl*)  
**ultimately** **show**  $\exists T'. T' \in T P' \wedge T \mapsto \text{Target}^* T'$   
 by *blast*  
**next**  
 case (*Suc n P''*)  
**assume**  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^{\text{Suc } n} P''$   
**from** *this* **obtain**  $P'$  **where**  $A1: \text{TargetTerm } T \mapsto (\text{STCal Source Target})^n P'$   
**and**  $A2: P' \mapsto (\text{STCal Source Target}) P''$   
 by *auto*

```

assume  $\bigwedge P'. \text{TargetTerm } T \mapsto (\text{STCal Source Target})^n P' \implies \exists T'. T' \in T P' \wedge T \mapsto \text{Target* } T'$ 
with  $A1$  obtain  $T'$  where  $A3: T' \in T P'$  and  $A4: T \mapsto \text{Target* } T'$ 
  by blast
from  $A2 A3$  obtain  $T''$  where  $A5: T'' \in T P''$  and  $A6: T' \mapsto \text{Target } T''$ 
  using STCal-step(2)[where  $T=T'$  and  $P'=P''$ ]
  by blast
from  $A4 A6$  have  $T \mapsto \text{Target* } T''$ 
  using step-to-steps[where  $\text{Cal}=\text{Target}$  and  $P=T'$  and  $P'=T''$ ]
  by (simp add: steps-add[where  $\text{Cal}=\text{Target}$  and  $P=T$  and  $Q=T'$  and  $R=T''$ ])
with  $A5$  show  $\exists T''. T'' \in T P'' \wedge T \mapsto \text{Target* } T''$ 
  by blast
qed
next
fix  $T'$ 
assume  $T \mapsto \text{Target* } T'$ 
from this obtain  $n$  where  $T \mapsto \text{Target}^n T'$ 
  by (auto simp add: steps-def)
thus  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T')$ 
proof (induct n arbitrary: T')
  case 0
  assume  $T \mapsto \text{Target}^0 T'$ 
  hence  $T = T'$ 
  by auto
  thus  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T')$ 
  by (simp add: steps-refl)
next
case (Suc n T'')
assume  $T \mapsto \text{Target}^{\text{Suc } n} T''$ 
from this obtain  $T'$  where  $B1: T \mapsto \text{Target}^n T'$  and  $B2: T' \mapsto \text{Target } T''$ 
  by auto
assume  $\bigwedge T'. T \mapsto \text{Target}^n T' \implies \text{TargetTerm } T \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T')$ 
with  $B1$  have  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T')$ 
  by blast
moreover from  $B2$  have  $\text{TargetTerm } T' \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T'')$ 
  using step-to-steps[where  $\text{Cal}=\text{STCal Source Target}$  and  $P=\text{TargetTerm } T'$ ]
  by (simp add: STCal-def)
ultimately show  $\text{TargetTerm } T \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T'')$ 
  by (rule steps-add)
qed
qed

lemma stepsST-STCal-steps:
fixes  $P P' :: ('procS, 'procT) Proc$ 
shows  $P \mapsto (\text{STCal Source Target})^* P' = P \mapsto \text{ST* } P'$ 
proof (cases P)
  case (SourceTerm SP)
  assume  $SP \in S P$ 
  thus  $P \mapsto (\text{STCal Source Target})^* P' = P \mapsto \text{ST* } P'$ 
  using STCal-steps(1)[where  $S=SP$  and  $P'=P'$ ] STSteps-steps(1)[where  $S=SP$  and  $P'=P'$ ]
  by blast
next
case (TargetTerm TP)
assume  $TP \in T P$ 
thus  $P \mapsto (\text{STCal Source Target})^* P' = P \mapsto \text{ST* } P'$ 
  using STCal-steps(2)[where  $T=TP$  and  $P'=P'$ ] STSteps-steps(2)[where  $T=TP$  and  $P'=P'$ ]
  by blast
qed

lemma stepsST-refl:
fixes  $P :: ('procS, 'procT) Proc$ 

```

**shows**  $P \mapsto ST^* P$   
**by** (*cases P, simp-all add: steps-refl*)

**lemma** *stepsST-add*:  
**fixes**  $P Q R :: ('procS, 'procT) Proc$   
**assumes**  $A1: P \mapsto ST^* Q$   
**and**  $A2: Q \mapsto ST^* R$   
**shows**  $P \mapsto ST^* R$

**proof** –  
**from**  $A1$  **have**  $P \mapsto (STCal Source Target)^* Q$   
**by** (*simp add: stepsST-STCal-steps*)  
**moreover from**  $A2$  **have**  $Q \mapsto (STCal Source Target)^* R$   
**by** (*simp add: stepsST-STCal-steps*)  
**ultimately have**  $P \mapsto (STCal Source Target)^* R$   
**by** (*rule steps-add*)  
**thus**  $P \mapsto ST^* R$   
**by** (*simp add: stepsST-STCal-steps*)  
**qed**

A divergent term of Proc is either a divergent source term or a divergent target term.

**abbreviation** *divergentST*  
 $:: ('procS, 'procT) Proc \Rightarrow bool \ (- \mapsto ST\omega [70] 80)$   
**where**  
 $P \mapsto ST\omega \equiv (\exists S. S \in S P \wedge S \mapsto (Source)\omega) \vee (\exists T. T \in T P \wedge T \mapsto (Target)\omega)$

**lemma** *STCal-divergent*:  
**fixes**  $S :: 'procS$   
**and**  $T :: 'procT$   
**shows**  $SourceTerm S \mapsto (STCal Source Target)\omega = S \mapsto (Source)\omega$   
**and**  $TargetTerm T \mapsto (STCal Source Target)\omega = T \mapsto (Target)\omega$   
**using** *STCal-steps*  
**by** (*auto simp add: STCal-def divergent-def*)

**lemma** *divergentST-STCal-divergent*:  
**fixes**  $P :: ('procS, 'procT) Proc$   
**shows**  $P \mapsto (STCal Source Target)\omega = P \mapsto ST\omega$   
**proof** (*cases P*)  
**case** (*SourceTerm SP*)  
**assume**  $SP \in S P$   
**thus**  $P \mapsto (STCal Source Target)\omega = P \mapsto ST\omega$   
**using** *STCal-divergent(1)*  
**by** *simp*  
**next**  
**case** (*TargetTerm TP*)  
**assume**  $TP \in T P$   
**thus**  $P \mapsto (STCal Source Target)\omega = P \mapsto ST\omega$   
**using** *STCal-divergent(2)*  
**by** *simp*  
**qed**

Similar to relations we define what it means for an encoding to preserve, reflect, or respect a predicate. An encoding preserves some predicate P if P(S) implies P(enc S) for all source terms S.

**abbreviation** *enc-preserves-pred*  $:: (('procS, 'procT) Proc \Rightarrow bool) \Rightarrow bool$  **where**  
 $enc-preserves-pred Pred \equiv \forall S. Pred (SourceTerm S) \longrightarrow Pred (TargetTerm (\llbracket S \rrbracket))$

**abbreviation** *enc-preserves-binary-pred*  
 $:: (('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$   
**where**  
 $enc-preserves-binary-pred Pred \equiv \forall S x. Pred (SourceTerm S) x \longrightarrow Pred (TargetTerm (\llbracket S \rrbracket)) x$

An encoding reflects some predicate P if P(S) implies P(enc S) for all source terms S.

**abbreviation** *enc-reflects-pred* :: (('procS, 'procT) Proc ⇒ bool) ⇒ bool **where**  
*enc-reflects-pred* Pred ≡ ∀ S. Pred (TargetTerm (⟦S⟧)) ⟶ Pred (SourceTerm S)

**abbreviation** *enc-reflects-binary-pred*  
:: (('procS, 'procT) Proc ⇒ 'b ⇒ bool) ⇒ bool  
**where**  
*enc-reflects-binary-pred* Pred ≡ ∀ S x. Pred (TargetTerm (⟦S⟧)) x ⟶ Pred (SourceTerm S) x

An encoding respects a predicate if it preserves and reflects it.

**abbreviation** *enc-respects-pred* :: (('procS, 'procT) Proc ⇒ bool) ⇒ bool **where**  
*enc-respects-pred* Pred ≡ *enc-preserves-pred* Pred ∧ *enc-reflects-pred* Pred

**abbreviation** *enc-respects-binary-pred*  
:: (('procS, 'procT) Proc ⇒ 'b ⇒ bool) ⇒ bool  
**where**  
*enc-respects-binary-pred* Pred ≡  
*enc-preserves-binary-pred* Pred ∧ *enc-reflects-binary-pred* Pred

**end**

To compare source terms and target terms w.r.t. their barbs or observables we assume that each languages defines its own predicate for the existence of barbs.

**locale** *encoding-wrt-barbs* =  
*encoding* Source Target Enc  
**for** Source :: 'procS processCalculus  
**and** Target :: 'procT processCalculus  
**and** Enc :: 'procS ⇒ 'procT +  
**fixes** SWB :: ('procS, 'barbs) calculusWithBarbs  
**and** TWB :: ('procT, 'barbs) calculusWithBarbs  
**assumes** calS: calculusWithBarbs.Calculus SWB = Source  
**and** calT: calculusWithBarbs.Calculus TWB = Target  
**begin**

**lemma** *STCalWB-STCal*:  
**shows** Calculus (STCalWB SWB TWB) = STCal Source Target  
**unfolding** *STCalWB-def* **using** calS calT  
**by** auto

We say a term P of Proc has some barbs a if either P is a source term that has barb a or P is a target term that has the barb b. For simplicity we assume that the sets of barbs is large enough to contain all barbs of the source terms, the target terms, and all barbs they might have in common.

**abbreviation** *hasBarbST*  
:: ('procS, 'procT) Proc ⇒ 'barbs ⇒ bool (-↓.- [70, 70] 80)  
**where**  
*P*↓.a ≡ (∃ S. S ∈ S P ∧ S↓<SWB>a) ∨ (∃ T. T ∈ T P ∧ T↓<TWB>a)

**lemma** *STCalWB-hasBarbST*:  
**fixes** P :: ('procS, 'procT) Proc  
**and** a :: 'barbs  
**shows** P↓<STCalWB SWB TWB>a = P↓.a  
**by** (*simp add: STCalWB-def*)

**lemma** *preservation-of-barbs-in-barbed-encoding*:  
**fixes** Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set  
**and** P Q :: ('procS, 'procT) Proc  
**and** a :: 'barbs  
**assumes** *preservation: rel-preserves-barbs* Rel (STCalWB SWB TWB)  
**and** *rel*: (P, Q) ∈ Rel  
**and** *barb*: P↓.a  
**shows** Q↓.a

using *preservation rel barb*  
 by (*simp add: STCalWB-def*)

**lemma** *reflection-of-barbs-in-barbed-encoding*:  
 fixes  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$  set  
 and  $P Q :: ('procS, 'procT) Proc$   
 and  $a :: 'barbs$   
 assumes *reflection: rel-reflects-barbs*  $Rel$  (*STCalWB SWB TWB*)  
 and *rel*:  $(P, Q) \in Rel$   
 and *barb*:  $Q \Downarrow a$   
 shows  $P \Downarrow a$   
 using *reflection rel barb*  
 by (*simp add: STCalWB-def*)

**lemma** *respection-of-barbs-in-barbed-encoding*:  
 fixes  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$  set  
 and  $P Q :: ('procS, 'procT) Proc$   
 and  $a :: 'barbs$   
 assumes *respection: rel-respects-barbs*  $Rel$  (*STCalWB SWB TWB*)  
 and *rel*:  $(P, Q) \in Rel$   
 shows  $P \Downarrow a = Q \Downarrow a$   
 using *preservation-of-barbs-in-barbed-encoding*[**where**  $Rel=Rel$  and  $P=P$  and  $Q=Q$  and  $a=a$ ]  
*reflection-of-barbs-in-barbed-encoding*[**where**  $Rel=Rel$  and  $P=P$  and  $Q=Q$  and  $a=a$ ]  
*respection rel*  
 by *blast*

A term  $P$  of  $Proc$  reaches a barb  $a$  if either  $P$  is a source term that reaches  $a$  or  $P$  is a target term that reaches  $a$ .

**abbreviation** *reachesBarbST*  
 $:: ('procS, 'procT) Proc \Rightarrow 'barbs \Rightarrow bool$  ( $-\Downarrow.- [70, 70] 80$ )  
 where  
 $P \Downarrow a \equiv (\exists S. S \in S \wedge S \Downarrow \langle SWB \rangle a) \vee (\exists T. T \in T \wedge T \Downarrow \langle TWB \rangle a)$

**lemma** *STCalWB-reachesBarbST*:  
 fixes  $P :: ('procS, 'procT) Proc$   
 and  $a :: 'barbs$   
 shows  $P \Downarrow \langle STCalWB SWB TWB \rangle a = P \Downarrow a$   
**proof** –  
 have  $\forall S. SourceTerm S \Downarrow \langle STCalWB SWB TWB \rangle a = SourceTerm S \Downarrow a$   
 using *STCal-steps(1)*  
 by (*auto simp add: STCalWB-def calS calT*)  
 moreover have  $\forall T. TargetTerm T \Downarrow \langle STCalWB SWB TWB \rangle a = TargetTerm T \Downarrow a$   
 using *STCal-steps(2)*  
 by (*auto simp add: STCalWB-def calS calT*)  
 ultimately show  $P \Downarrow \langle STCalWB SWB TWB \rangle a = P \Downarrow a$   
 by (*cases P, simp+*)  
**qed**

**lemma** *weak-preservation-of-barbs-in-barbed-encoding*:  
 fixes  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$  set  
 and  $P Q :: ('procS, 'procT) Proc$   
 and  $a :: 'barbs$   
 assumes *preservation: rel-weakly-preserves-barbs*  $Rel$  (*STCalWB SWB TWB*)  
 and *rel*:  $(P, Q) \in Rel$   
 and *barb*:  $P \Downarrow a$   
 shows  $Q \Downarrow a$   
**proof** –  
 from *barb* have  $P \Downarrow \langle STCalWB SWB TWB \rangle a$   
 by (*simp add: STCalWB-reachesBarbST*)  
 with *preservation rel* have  $Q \Downarrow \langle STCalWB SWB TWB \rangle a$

```

  by blast
thus  $Q \Downarrow.a$ 
  by (simp add: STCalWB-reachesBarbST)
qed

lemma weak-reflection-of-barbs-in-barbed-encoding:
  fixes  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$  set
  and  $P Q :: ('procS, 'procT) Proc$ 
  and  $a :: 'barbs$ 
  assumes reflection: rel-weakly-reflects-barbs  $Rel$  (STCalWB SWB TWB)
  and rel:  $(P, Q) \in Rel$ 
  and barb:  $Q \Downarrow.a$ 
  shows  $P \Downarrow.a$ 
proof -
  from barb have  $Q \Downarrow <STCalWB SWB TWB> a$ 
  by (simp add: STCalWB-reachesBarbST)
  with reflection rel have  $P \Downarrow <STCalWB SWB TWB> a$ 
  by blast
  thus  $P \Downarrow.a$ 
  by (simp add: STCalWB-reachesBarbST)
qed

lemma weak-respection-of-barbs-in-barbed-encoding:
  fixes  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$  set
  and  $P Q :: ('procS, 'procT) Proc$ 
  and  $a :: 'barbs$ 
  assumes respection: rel-weakly-respects-barbs  $Rel$  (STCalWB SWB TWB)
  and rel:  $(P, Q) \in Rel$ 
  shows  $P \Downarrow.a = Q \Downarrow.a$ 
proof (rule iffI)
  assume  $P \Downarrow.a$ 
  with respection rel show  $Q \Downarrow.a$ 
  using weak-preservation-of-barbs-in-barbed-encoding[where  $Rel=Rel$ ]
  by blast
next
  assume  $Q \Downarrow.a$ 
  with respection rel show  $P \Downarrow.a$ 
  using weak-reflection-of-barbs-in-barbed-encoding[where  $Rel=Rel$ ]
  by blast
qed

end

end

theory SourceTargetRelation
  imports Encodings SimulationRelations
begin

```

## 5 Relation between Source and Target Terms

### 5.1 Relations Induced by the Encoding Function

We map encodability criteria on conditions of relations between source and target terms. The encoding function itself induces such relations. To analyse the preservation of source term behaviours we use relations that contain the pairs  $(S, \text{enc } S)$  for all source terms  $S$ .

```

inductive-set (in encoding) indRelR
  :: (((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set
  where
  encR: (SourceTerm  $S$ , TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  indRelR

```

**abbreviation** (in *encoding*) *indRelRinfix* ::  
 ('procS, 'procT) Proc  $\Rightarrow$  ('procS, 'procT) Proc  $\Rightarrow$  bool (-  $\mathcal{R}[\cdot]R$  - [75, 75] 80)  
**where**  
 P  $\mathcal{R}[\cdot]R$  Q  $\equiv$  (P, Q)  $\in$  *indRelR*

**inductive-set** (in *encoding*) *indRelRPO*  
 :: (((('procS, 'procT) Proc)  $\times$  (('procS, 'procT) Proc)) set  
**where**  
*encR*: (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  *indRelRPO* |  
*source*: (SourceTerm S, SourceTerm S)  $\in$  *indRelRPO* |  
*target*: (TargetTerm T, TargetTerm T)  $\in$  *indRelRPO* |  
*trans*:  $\llbracket (P, Q) \in \textit{indRelRPO}; (Q, R) \in \textit{indRelRPO} \rrbracket \Longrightarrow (P, R) \in \textit{indRelRPO}$

**abbreviation** (in *encoding*) *indRelRPOinfix* ::  
 ('procS, 'procT) Proc  $\Rightarrow$  ('procS, 'procT) Proc  $\Rightarrow$  bool (-  $\lesssim[\cdot]R$  - [75, 75] 80)  
**where**  
 P  $\lesssim[\cdot]R$  Q  $\equiv$  (P, Q)  $\in$  *indRelRPO*

**lemma** (in *encoding*) *indRelRPO-refl*:

**shows** *refl indRelRPO*  
**unfolding** *refl-on-def*  
**proof** *auto*  
**fix** P  
**show** P  $\lesssim[\cdot]R$  P  
**proof** (*cases* P)  
**case** (SourceTerm SP)  
**assume** SP  $\in$  S P  
**thus** P  $\lesssim[\cdot]R$  P  
**by** (*simp add: indRelRPO.source*)  
**next**  
**case** (TargetTerm TP)  
**assume** TP  $\in$  T P  
**thus** P  $\lesssim[\cdot]R$  P  
**by** (*simp add: indRelRPO.target*)  
**qed**  
**qed**

**lemma** (in *encoding*) *indRelRPO-is-preorder*:

**shows** *preorder indRelRPO*  
**unfolding** *preorder-on-def*  
**proof**  
**show** *refl indRelRPO*  
**by** (*rule indRelRPO-refl*)  
**next**  
**show** *trans indRelRPO*  
**unfolding** *trans-def*  
**proof** *clarify*  
**fix** P Q R  
**assume** P  $\lesssim[\cdot]R$  Q and Q  $\lesssim[\cdot]R$  R  
**thus** P  $\lesssim[\cdot]R$  R  
**by** (*rule indRelRPO.trans*)  
**qed**  
**qed**

**lemma** (in *encoding*) *refl-trans-closure-of-indRelR*:

**shows** *indRelRPO = indRelR\**  
**proof** *auto*  
**fix** P Q  
**assume** P  $\lesssim[\cdot]R$  Q  
**thus** (P, Q)  $\in$  *indRelR\**

```

proof induct
  case (encR S)
  show (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  indRelR*
    using indRelR.encR[of S]
    by simp
next
  case (source S)
  show (SourceTerm S, SourceTerm S)  $\in$  indRelR*
    by simp
next
  case (target T)
  show (TargetTerm T, TargetTerm T)  $\in$  indRelR*
    by simp
next
  case (trans P Q R)
  assume (P, Q)  $\in$  indRelR* and (Q, R)  $\in$  indRelR*
  thus (P, R)  $\in$  indRelR*
    by simp
qed
next
fix P Q
assume (P, Q)  $\in$  indRelR*
thus  $P \lesssim[\cdot]R Q$ 
proof induct
  show  $P \lesssim[\cdot]R P$ 
    using indRelRPO-refl
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume  $P \lesssim[\cdot]R Q$ 
  moreover assume  $Q \mathcal{R}[\cdot]R R$ 
  hence  $Q \lesssim[\cdot]R R$ 
    by (induct, simp add: indRelRPO.encR)
  ultimately show  $P \lesssim[\cdot]R R$ 
    by (rule indRelRPO.trans)
qed
qed

```

The relation *indRelR* is the smallest relation that relates all source terms and their literal translations. Thus there exists a relation that relates source terms and their literal translations and satisfies some predicate on its pairs iff the predicate holds for the pairs of *indRelR*.

**lemma** (*in encoding*) *indRelR-impl-exists-source-target-relation*:

```

fixes PredA :: (('procS, 'procT) Proc  $\times$  ('procS, 'procT) Proc) set  $\Rightarrow$  bool
  and PredB :: (('procS, 'procT) Proc  $\times$  ('procS, 'procT) Proc)  $\Rightarrow$  bool
shows PredA indRelR  $\Longrightarrow$   $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)  $\wedge$  PredA Rel
  and  $\forall (P, Q) \in$  indRelR. PredB (P, Q)
     $\Longrightarrow$   $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)  $\wedge$  ( $\forall (P, Q) \in$  Rel. PredB (P, Q))
proof –
  have A:  $\forall S$ . SourceTerm S  $\mathcal{R}[\cdot]R$  TargetTerm ( $\llbracket S \rrbracket$ )
    by (simp add: indRelR.encR)
  thus PredA indRelR  $\Longrightarrow$   $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)  $\wedge$  PredA Rel
    by blast
  with A show  $\forall (P, Q) \in$  indRelR. PredB (P, Q)
     $\Longrightarrow$   $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)  $\wedge$  ( $\forall (P, Q) \in$  Rel. PredB (P, Q))
    by blast
qed

```

**lemma** (*in encoding*) *source-target-relation-impl-indRelR*:

```

fixes Rel :: (('procS, 'procT) Proc  $\times$  ('procS, 'procT) Proc) set

```



```

and Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
assumes encRRel: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
and condRel: ∀ (P, Q) ∈ Rel. Pred (P, Q)
shows ∀ (P, Q) ∈ indRelR. Pred (P, Q)
proof clarify
  fix P Q
  assume P  $\mathcal{R}$ [·]R Q
  with encRRel have (P, Q) ∈ Rel
    by (auto simp add: indRelR.simps)
  with condRel show Pred (P, Q)
    by simp
qed

```

```

lemma (in encoding) indRelR-iff-exists-source-target-relation:
fixes Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
shows (∀ (P, Q) ∈ indRelR. Pred (P, Q))
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)))
using indRelR-impl-exists-source-target-relation(2)[where PredB=Pred]
  source-target-relation-impl-indRelR[where Pred=Pred]
by blast

```

```

lemma (in encoding) indRelR-modulo-pred-impl-indRelRPO-modulo-pred:
fixes Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
assumes reflCond: ∀ P. Pred (P, P)
and transCond: ∀ P Q R. Pred (P, Q) ∧ Pred (Q, R) ⟶ Pred (P, R)
shows (∀ (P, Q) ∈ indRelR. Pred (P, Q)) = (∀ (P, Q) ∈ indRelRPO. Pred (P, Q))
proof auto
  fix P Q
  assume A: ∀ x ∈ indRelR. Pred x
  assume P  $\lesssim$ [·]R Q
  thus Pred (P, Q)
  proof induct
    case (encR S)
    have SourceTerm S  $\mathcal{R}$ [·]R TargetTerm (⟦S⟧)
      by (simp add: indRelR.encR)
    with A show Pred (SourceTerm S, TargetTerm (⟦S⟧))
      by simp
    next
    case (source S)
    from reflCond show Pred (SourceTerm S, SourceTerm S)
      by simp
    next
    case (target T)
    from reflCond show Pred (TargetTerm T, TargetTerm T)
      by simp
    next
    case (trans P Q R)
    assume Pred (P, Q) and Pred (Q, R)
    with transCond show Pred (P, R)
      by blast
  qed
next
  fix P Q
  assume ∀ x ∈ indRelRPO. Pred x and P  $\mathcal{R}$ [·]R Q
  thus Pred (P, Q)
    by (auto simp add: indRelRPO.encR indRelR.simps)
qed

```

```

lemma (in encoding) indRelRPO-iff-exists-source-target-relation:
fixes Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
shows (∀ (P, Q) ∈ indRelRPO. Pred (P, Q)) = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)

```

```

     $\wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder Rel}$ 
proof (rule iffI)
  have  $\forall S. \text{SourceTerm } S \lesssim [\cdot] R \text{TargetTerm } (\llbracket S \rrbracket)$ 
    by (simp add: indRelRPO.encR)
  moreover have preorder indRelRPO
    using indRelRPO-is-preorder
    by blast
  moreover assume  $\forall (P, Q) \in \text{indRelRPO}. \text{Pred } (P, Q)$ 
  ultimately show  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$ 
     $\wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder Rel}$ 
    by blast
next
  assume  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$ 
     $\wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder Rel}$ 
  from this obtain Rel where A1:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$ 
    and A2:  $\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)$  and A3: preorder Rel
    by blast
  show  $\forall (P, Q) \in \text{indRelRPO}. \text{Pred } (P, Q)$ 
proof clarify
  fix P Q
  assume  $P \lesssim [\cdot] R Q$ 
  hence  $(P, Q) \in \text{Rel}$ 
proof induct
  case (encR S)
  from A1 show  $(\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$ 
    by simp
next
  case (source S)
  from A3 show  $(\text{SourceTerm } S, \text{SourceTerm } S) \in \text{Rel}$ 
    unfolding preorder-on-def refl-on-def
    by simp
next
  case (target T)
  from A3 show  $(\text{TargetTerm } T, \text{TargetTerm } T) \in \text{Rel}$ 
    unfolding preorder-on-def refl-on-def
    by simp
next
  case (trans P Q R)
  assume  $(P, Q) \in \text{Rel}$  and  $(Q, R) \in \text{Rel}$ 
  with A3 show  $(P, R) \in \text{Rel}$ 
    unfolding preorder-on-def trans-def
    by blast
qed
with A2 show  $\text{Pred } (P, Q)$ 
  by simp
qed
qed

```

An encoding preserves, reflects, or respects a predicate iff indRelR preserves, reflects, or respects this predicate.

**lemma** (*in encoding*) *enc-satisfies-pred-impl-indRelR-satisfies-pred*:  
**fixes** *Pred* ::  $((\text{'procS}, \text{'procT}) \text{Proc} \times (\text{'procS}, \text{'procT}) \text{Proc}) \Rightarrow \text{bool}$   
**assumes** *encCond*:  $\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))$   
**shows**  $\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q)$   
**by** (*auto simp add: encCond indRelR.simps*)

**lemma** (*in encoding*) *indRelR-satisfies-pred-impl-enc-satisfies-pred*:  
**fixes** *Pred* ::  $((\text{'procS}, \text{'procT}) \text{Proc} \times (\text{'procS}, \text{'procT}) \text{Proc}) \Rightarrow \text{bool}$   
**assumes** *relCond*:  $\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q)$   
**shows**  $\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))$

using *relCond indRelR.encR*  
 by *simp*

**lemma** (in *encoding*) *enc-satisfies-pred-iff-indRelR-satisfies-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $(\forall S. Pred (SourceTerm S, TargetTerm (\llbracket S \rrbracket))) = (\forall (P, Q) \in indRelR. Pred (P, Q))$   
 using *enc-satisfies-pred-impl-indRelR-satisfies-pred*[**where**  $Pred = Pred$ ]  
*indRelR-satisfies-pred-impl-enc-satisfies-pred*[**where**  $Pred = Pred$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
 shows  $(\forall S a. Pred (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) a) = (\forall (P, Q) \in indRelR. \forall a. Pred (P, Q) a)$   
 using *enc-satisfies-pred-iff-indRelR-satisfies-pred*  
 by *simp*

**lemma** (in *encoding*) *enc-preserves-pred-iff-indRelR-preserves-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $enc\text{-preserves-pred } Pred = rel\text{-preserves-pred } indRelR\ Pred$   
 using *enc-satisfies-pred-iff-indRelR-satisfies-pred*[**where**  $Pred = \lambda(P, Q). Pred P \longrightarrow Pred Q$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-preserves-binary-pred-iff-indRelR-preserves-binary-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
 shows  $enc\text{-preserves-binary-pred } Pred = rel\text{-preserves-binary-pred } indRelR\ Pred$   
 using *enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred*[**where**  
 $Pred = \lambda(P, Q) a. Pred P a \longrightarrow Pred Q a$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-preserves-pred-iff-indRelRPO-preserves-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $enc\text{-preserves-pred } Pred = rel\text{-preserves-pred } indRelRPO\ Pred$   
 using *enc-preserves-pred-iff-indRelR-preserves-pred*[**where**  $Pred = Pred$ ]  
*indRelR-modulo-pred-impl-indRelRPO-modulo-pred*[**where**  
 $Pred = \lambda(P, Q). Pred P \longrightarrow Pred Q$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-reflects-pred-iff-indRelR-reflects-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $enc\text{-reflects-pred } Pred = rel\text{-reflects-pred } indRelR\ Pred$   
 using *enc-satisfies-pred-iff-indRelR-satisfies-pred*[**where**  $Pred = \lambda(P, Q). Pred Q \longrightarrow Pred P$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
 shows  $enc\text{-reflects-binary-pred } Pred = rel\text{-reflects-binary-pred } indRelR\ Pred$   
 using *enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred*[**where**  
 $Pred = \lambda(P, Q) a. Pred Q a \longrightarrow Pred P a$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-reflects-pred-iff-indRelRPO-reflects-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $enc\text{-reflects-pred } Pred = rel\text{-reflects-pred } indRelRPO\ Pred$   
 using *enc-reflects-pred-iff-indRelR-reflects-pred*[**where**  $Pred = Pred$ ]  
*indRelR-modulo-pred-impl-indRelRPO-modulo-pred*[**where**  
 $Pred = \lambda(P, Q). Pred Q \longrightarrow Pred P$ ]  
 by *blast*

**lemma** (in *encoding*) *enc-respects-pred-iff-indRelR-respects-pred*:  
 fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
 shows  $enc\text{-respects-pred } Pred = rel\text{-respects-pred } indRelR\ Pred$

using *enc-preserves-pred-iff-indRelR-preserves-pred*[**where** *Pred=Pred*]  
*enc-reflects-pred-iff-indRelR-reflects-pred*[**where** *Pred=Pred*]  
 by *blast*

**lemma** (*in encoding*) *enc-respects-binary-pred-iff-indRelR-respects-binary-pred*:  
**fixes** *Pred* :: ('procS, 'procT) Proc  $\Rightarrow$  'b  $\Rightarrow$  bool  
**shows** *enc-respects-binary-pred* *Pred* = *rel-respects-binary-pred* *indRelR* *Pred*  
 using *enc-preserves-binary-pred-iff-indRelR-preserves-binary-pred*[**where** *Pred=Pred*]  
*enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred*[**where** *Pred=Pred*]  
 by *blast*

**lemma** (*in encoding*) *enc-respects-pred-iff-indRelRPO-respects-pred*:  
**fixes** *Pred* :: ('procS, 'procT) Proc  $\Rightarrow$  bool  
**shows** *enc-respects-pred* *Pred* = *rel-respects-pred* *indRelRPO* *Pred*  
 using *enc-respects-pred-iff-indRelR-respects-pred*[**where** *Pred=Pred*]  
*indRelR-modulo-pred-impl-indRelRPO-modulo-pred*[**where** *Pred*= $\lambda(P, Q). \text{Pred } Q = \text{Pred } P$ ]  
 apply *simp* by *blast*

Accordingly an encoding preserves, reflects, or respects a predicate iff there exists a relation that relates source terms with their literal translations and preserves, reflects, or respects this predicate.

**lemma** (*in encoding*) *enc-satisfies-pred-iff-source-target-satisfies-pred*:  
**fixes** *Pred* :: (('procS, 'procT) Proc  $\times$  ('procS, 'procT) Proc)  $\Rightarrow$  bool  
**shows**  $(\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$   
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)))$   
**and**  $\llbracket \forall P Q R. \text{Pred } (P, Q) \wedge \text{Pred } (Q, R) \longrightarrow \text{Pred } (P, R); \forall P. \text{Pred } (P, P) \rrbracket \Longrightarrow$   
 $(\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))) = (\exists \text{Rel}. (\forall S.$   
 $(\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder } \text{Rel})$

**proof** –  
**show**  $(\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$   
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)))$   
 using *enc-satisfies-pred-iff-indRelR-satisfies-pred*[**where** *Pred=Pred*]  
*indRelR-iff-exists-source-target-relation*[**where** *Pred=Pred*]  
 by *simp*

**next**  
**have**  $(\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))) = (\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q))$   
 using *enc-satisfies-pred-iff-indRelR-satisfies-pred*[**where** *Pred=Pred*]  
 by *simp*  
**moreover assume**  $\forall P Q R. \text{Pred } (P, Q) \wedge \text{Pred } (Q, R) \longrightarrow \text{Pred } (P, R)$  **and**  $\forall P. \text{Pred } (P, P)$   
**hence**  $(\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q)) = (\forall (P, Q) \in \text{indRelRPO}. \text{Pred } (P, Q))$   
 using *indRelR-modulo-pred-impl-indRelRPO-modulo-pred*[**where** *Pred=Pred*]  
 by *blast*  
**ultimately show**  $(\forall S. \text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))) = (\exists \text{Rel}.$   
 $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder } \text{Rel})$   
 using *indRelRPO-iff-exists-source-target-relation*[**where** *Pred=Pred*]  
 by *simp*  
**qed**

**lemma** (*in encoding*) *enc-preserves-pred-iff-source-target-rel-preserves-pred*:  
**fixes** *Pred* :: ('procS, 'procT) Proc  $\Rightarrow$  bool  
**shows** *enc-preserves-pred* *Pred*  
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge \text{rel-preserves-pred } \text{Rel } \text{Pred})$   
**and** *enc-preserves-pred* *Pred* =  $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{rel-preserves-pred } \text{Rel } \text{Pred} \wedge \text{preorder } \text{Rel})$

**proof** –  
**have** *A1*: *enc-preserves-pred* *Pred*  
 $= (\forall S. (\lambda(P, Q). \text{Pred } P \longrightarrow \text{Pred } Q) (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$   
 by *blast*  
**moreover have** *A2*:  $\bigwedge \text{Rel}. \text{rel-preserves-pred } \text{Rel } \text{Pred}$   
 $= (\forall (P, Q) \in \text{Rel}. (\lambda(P, Q). \text{Pred } P \longrightarrow \text{Pred } Q) (P, Q))$   
 by *blast*

**ultimately show**  $enc\text{-preserves-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ } Pred)$   
**using**  $enc\text{-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  
 $Pred = \lambda(P, Q). Pred P \longrightarrow Pred Q]$   
**by simp**  
**from**  $A1 A2$  **show**  $enc\text{-preserves-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ } Pred \wedge preorder \text{ } Rel)$   
**using**  $enc\text{-satisfies-pred-iff-source-target-satisfies-pred}(2)$ **[where**  
 $Pred = \lambda(P, Q). Pred P \longrightarrow Pred Q]$   
**by simp**  
**qed**

**lemma** (**in encoding**)  $enc\text{-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred}$ :  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
**shows**  $enc\text{-preserves-binary-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ } Pred)$

**proof** –

**have**  $enc\text{-preserves-binary-pred } Pred$   
 $= (\forall S. (\lambda(P, Q). \forall a. Pred P a \longrightarrow Pred Q a) (SourceTerm S, TargetTerm (\llbracket S \rrbracket)))$   
**by blast**

**moreover have**  $\bigwedge Rel. rel\text{-preserves-binary-pred } Rel \text{ } Pred$   
 $= (\forall (P, Q) \in Rel. (\lambda(P, Q). \forall a. Pred P a \longrightarrow Pred Q a) (P, Q))$

**by blast**

**ultimately show**  $enc\text{-preserves-binary-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ } Pred)$

**using**  $enc\text{-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  
 $Pred = \lambda(P, Q). \forall a. Pred P a \longrightarrow Pred Q a]$

**by simp**

**qed**

**lemma** (**in encoding**)  $enc\text{-reflects-pred-iff-source-target-rel-reflects-pred}$ :

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$

**shows**  $enc\text{-reflects-pred } Pred$

$= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ } Pred)$

**and**  $enc\text{-reflects-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ } Pred \wedge preorder \text{ } Rel)$

**proof** –

**have**  $A1: enc\text{-reflects-pred } Pred$   
 $= (\forall S. (\lambda(P, Q). Pred Q \longrightarrow Pred P) (SourceTerm S, TargetTerm (\llbracket S \rrbracket)))$

**by blast**

**moreover have**  $A2: \bigwedge Rel. rel\text{-reflects-pred } Rel \text{ } Pred$   
 $= (\forall (P, Q) \in Rel. (\lambda(P, Q). Pred Q \longrightarrow Pred P) (P, Q))$

**by blast**

**ultimately show**  $enc\text{-reflects-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ } Pred)$

**using**  $enc\text{-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  
 $Pred = \lambda(P, Q). Pred Q \longrightarrow Pred P]$

**by simp**

**from**  $A1 A2$  **show**  $enc\text{-reflects-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ } Pred \wedge preorder \text{ } Rel)$

**using**  $enc\text{-satisfies-pred-iff-source-target-satisfies-pred}(2)$ **[where**  
 $Pred = \lambda(P, Q). Pred Q \longrightarrow Pred P]$

**by simp**

**qed**

**lemma** (**in encoding**)  $enc\text{-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred}$ :

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$

**shows**  $enc\text{-reflects-binary-pred } Pred = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ } Pred)$

**proof** –

**have**  $enc\text{-reflects-binary-pred } Pred$

$$= (\forall S. (\lambda(P, Q). \forall a. \text{Pred } Q \ a \longrightarrow \text{Pred } P \ a) (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$$
**by** *blast*  
**moreover have**  $\bigwedge \text{Rel. rel-reflects-binary-pred } \text{Rel } \text{Pred}$   

$$= (\forall (P, Q) \in \text{Rel. } (\lambda(P, Q). \forall a. \text{Pred } Q \ a \longrightarrow \text{Pred } P \ a) (P, Q))$$
**by** *blast*  
**ultimately show**  $\text{enc-reflects-binary-pred } \text{Pred} = (\exists \text{Rel. } (\forall S.$   

$$(\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge \text{rel-reflects-binary-pred } \text{Rel } \text{Pred})$$
**using**  $\text{enc-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  

$$\text{Pred} = \lambda(P, Q). \forall a. \text{Pred } Q \ a \longrightarrow \text{Pred } P \ a]$$
**by** *simp*  
**qed**

**lemma** (*in encoding*)  $\text{enc-respects-pred-iff-source-target-rel-respects-pred-encR}$ :  
**fixes**  $\text{Pred} :: ('procS, 'procT) \text{Proc} \Rightarrow \text{bool}$   
**shows**  $\text{enc-respects-pred } \text{Pred}$   

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge \text{rel-respects-pred } \text{Rel } \text{Pred})$$
**and**  $\text{enc-respects-pred } \text{Pred} = (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   

$$\wedge \text{rel-respects-pred } \text{Rel } \text{Pred} \wedge \text{preorder } \text{Rel})$$
**proof** –  
**have**  $A1: \text{enc-respects-pred } \text{Pred}$   

$$= (\forall S. (\lambda(P, Q). \text{Pred } P = \text{Pred } Q) (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$$
**by** *blast*  
**moreover**  
**have**  $A2: \bigwedge \text{Rel. rel-respects-pred } \text{Rel } \text{Pred} = (\forall (P, Q) \in \text{Rel. } (\lambda(P, Q). \text{Pred } P = \text{Pred } Q) (P, Q))$   
**by** *blast*  
**ultimately show**  $\text{enc-respects-pred } \text{Pred} = (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   

$$\wedge \text{rel-respects-pred } \text{Rel } \text{Pred})$$
**using**  $\text{enc-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  

$$\text{Pred} = \lambda(P, Q). \text{Pred } P = \text{Pred } Q]$$
**by** *simp*  
**from**  $A1 \ A2$  **show**  $\text{enc-respects-pred } \text{Pred} = (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   

$$\wedge \text{rel-respects-pred } \text{Rel } \text{Pred} \wedge \text{preorder } \text{Rel})$$
**using**  $\text{enc-satisfies-pred-iff-source-target-satisfies-pred}(2)$ **[where**  

$$\text{Pred} = \lambda(P, Q). \text{Pred } P = \text{Pred } Q]$$
**by** *simp*  
**qed**

**lemma** (*in encoding*)  $\text{enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR}$ :  
**fixes**  $\text{Pred} :: ('procS, 'procT) \text{Proc} \Rightarrow 'b \Rightarrow \text{bool}$   
**shows**  $\text{enc-respects-binary-pred } \text{Pred} = (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   

$$\wedge \text{rel-respects-binary-pred } \text{Rel } \text{Pred})$$
**proof** –  
**have**  $\text{enc-respects-binary-pred } \text{Pred}$   

$$= (\forall S. (\lambda(P, Q). \forall a. \text{Pred } P \ a = \text{Pred } Q \ a) (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)))$$
**by** *blast*  
**moreover have**  $\bigwedge \text{Rel. rel-respects-binary-pred } \text{Rel } \text{Pred}$   

$$= (\forall (P, Q) \in \text{Rel. } (\lambda(P, Q). \forall a. \text{Pred } P \ a = \text{Pred } Q \ a) (P, Q))$$
**by** *blast*  
**ultimately show**  $\text{enc-respects-binary-pred } \text{Pred} = (\exists \text{Rel. } (\forall S.$   

$$(\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge \text{rel-respects-binary-pred } \text{Rel } \text{Pred})$$
**using**  $\text{enc-satisfies-pred-iff-source-target-satisfies-pred}(1)$ **[where**  

$$\text{Pred} = \lambda(P, Q). \forall a. \text{Pred } P \ a = \text{Pred } Q \ a]$$
**by** *simp*  
**qed**

To analyse the reflection of source term behaviours we use relations that contain the pairs  $(\text{enc } S, S)$  for all source terms  $S$ .

**inductive-set** (*in encoding*)  $\text{indRelL}$   

$$:: (((\text{'procS}, \text{'procT}) \text{Proc}) \times ((\text{'procS}, \text{'procT}) \text{Proc})) \text{set}$$
**where**

*encL*: (*TargetTerm* ( $\llbracket S \rrbracket$ ), *SourceTerm* *S*)  $\in$  *indRelL*

**abbreviation** (in *encoding*) *indRelL*infix ::

(*'procS*, *'procT*) *Proc*  $\Rightarrow$  (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* (-  $\mathcal{R}[\cdot]L$  - [75, 75] 80)

**where**

$P \mathcal{R}[\cdot]L Q \equiv (P, Q) \in \textit{indRelL}$

**inductive-set** (in *encoding*) *indRelLPO*

:: (((*'procS*, *'procT*) *Proc*)  $\times$  (((*'procS*, *'procT*) *Proc*)) *set*

**where**

*encL*: (*TargetTerm* ( $\llbracket S \rrbracket$ ), *SourceTerm* *S*)  $\in$  *indRelLPO* |

*source*: (*SourceTerm* *S*, *SourceTerm* *S*)  $\in$  *indRelLPO* |

*target*: (*TargetTerm* *T*, *TargetTerm* *T*)  $\in$  *indRelLPO* |

*trans*:  $\llbracket (P, Q) \in \textit{indRelLPO}; (Q, R) \in \textit{indRelLPO} \rrbracket \Longrightarrow (P, R) \in \textit{indRelLPO}$

**abbreviation** (in *encoding*) *indRelLPO*infix ::

(*'procS*, *'procT*) *Proc*  $\Rightarrow$  (*'procS*, *'procT*) *Proc*  $\Rightarrow$  *bool* (-  $\lesssim[\cdot]L$  - [75, 75] 80)

**where**

$P \lesssim[\cdot]L Q \equiv (P, Q) \in \textit{indRelLPO}$

**lemma** (in *encoding*) *indRelLPO*-refl:

**shows** *refl* *indRelLPO*

**unfolding** *refl-on-def*

**proof** *auto*

**fix** *P*

**show**  $P \lesssim[\cdot]L P$

**proof** (*cases* *P*)

**case** (*SourceTerm* *SP*)

**assume**  $SP \in S P$

**thus**  $P \lesssim[\cdot]L P$

**by** (*simp* *add*: *indRelLPO.source*)

**next**

**case** (*TargetTerm* *TP*)

**assume**  $TP \in T P$

**thus**  $P \lesssim[\cdot]L P$

**by** (*simp* *add*: *indRelLPO.target*)

**qed**

**qed**

**lemma** (in *encoding*) *indRelLPO*-is-preorder:

**shows** *preorder* *indRelLPO*

**unfolding** *preorder-on-def*

**proof**

**show** *refl* *indRelLPO*

**by** (*rule* *indRelLPO-refl*)

**next**

**show** *trans* *indRelLPO*

**unfolding** *trans-def*

**proof** *clarify*

**fix** *P Q R*

**assume**  $P \lesssim[\cdot]L Q$  **and**  $Q \lesssim[\cdot]L R$

**thus**  $P \lesssim[\cdot]L R$

**by** (*rule* *indRelLPO.trans*)

**qed**

**qed**

**lemma** (in *encoding*) *refl-trans-closure-of-indRelL*:

**shows** *indRelLPO* = *indRelL*\*

**proof** *auto*

**fix** *P Q*

**assume**  $P \lesssim[\cdot]L Q$

```

thus (P, Q) ∈ indRelL*
proof induct
  case (encL S)
  show (TargetTerm ([S]), SourceTerm S) ∈ indRelL*
    using indRelL.encL[of S]
  by simp
next
  case (source S)
  show (SourceTerm S, SourceTerm S) ∈ indRelL*
    by simp
next
  case (target T)
  show (TargetTerm T, TargetTerm T) ∈ indRelL*
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ indRelL* and (Q, R) ∈ indRelL*
  thus (P, R) ∈ indRelL*
    by simp
qed
next
fix P Q
assume (P, Q) ∈ indRelL*
thus P ≲[·]L Q
proof induct
  show P ≲[·]L P
    using indRelLPO-refl
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume P ≲[·]L Q
  moreover assume Q ≲[·]L R
  hence P ≲[·]L R
    by (induct, simp add: indRelLPO.encL)
  ultimately show P ≲[·]L R
    by (simp add: indRelLPO.trans[of P Q R])
qed
qed

```

The relations  $\text{indRelR}$  and  $\text{indRelL}$  are dual.  $\text{indRelR}$  preserves some predicate iff  $\text{indRelL}$  reflects it.  $\text{indRelR}$  reflects some predicate iff  $\text{indRelL}$  reflects it.  $\text{indRelR}$  respects some predicate iff  $\text{indRelL}$  does.

```

lemma (in encoding) indRelR-preserves-pred-iff-indRelL-reflects-pred:
  fixes Pred :: ('procS, 'procT) Proc ⇒ bool
  shows rel-preserves-pred indRelR Pred = rel-reflects-pred indRelL Pred
proof
  assume preservation: rel-preserves-pred indRelR Pred
  show rel-reflects-pred indRelL Pred
proof clarify
  fix P Q
  assume P ≲[·]L Q
  from this obtain S where S ∈ S Q and [S] ∈ T P
    by (induct, blast)
  hence Q ≲[·]R P
    by (simp add: indRelR.encR)
  moreover assume Pred Q
  ultimately show Pred P
    using preservation
    by blast

```



```

qed
next
assume reflection: rel-reflects-pred indRelL Pred
show rel-preserves-pred indRelR Pred
proof clarify
  fix P Q
  assume P  $\mathcal{R}[\cdot]R$  Q
  from this obtain S where  $S \in S$  P and  $\llbracket S \rrbracket \in T$  Q
  by (induct, blast)
  hence Q  $\mathcal{R}[\cdot]L$  P
  by (simp add: indRelL.encL)
  moreover assume Pred P
  ultimately show Pred Q
  using reflection
  by blast
qed
qed

```

```

lemma (in encoding) indRelR-preserves-binary-pred-iff-indRelL-reflects-binary-pred:
  fixes Pred :: ('procS, 'procT) Proc  $\Rightarrow$  'b  $\Rightarrow$  bool
  shows rel-preserves-binary-pred indRelR Pred = rel-reflects-binary-pred indRelL Pred
proof

```

```

  assume preservation: rel-preserves-binary-pred indRelR Pred
  show rel-reflects-binary-pred indRelL Pred
proof clarify
  fix P Q x
  assume P  $\mathcal{R}[\cdot]L$  Q
  from this obtain S where  $S \in S$  Q and  $\llbracket S \rrbracket \in T$  P
  by (induct, blast)
  hence Q  $\mathcal{R}[\cdot]R$  P
  by (simp add: indRelR.encR)
  moreover assume Pred Q x
  ultimately show Pred P x
  using preservation
  by blast
qed

```

```

next
assume reflection: rel-reflects-binary-pred indRelL Pred
show rel-preserves-binary-pred indRelR Pred
proof clarify
  fix P Q x
  assume P  $\mathcal{R}[\cdot]R$  Q
  from this obtain S where  $S \in S$  P and  $\llbracket S \rrbracket \in T$  Q
  by (induct, blast)
  hence Q  $\mathcal{R}[\cdot]L$  P
  by (simp add: indRelL.encL)
  moreover assume Pred P x
  ultimately show Pred Q x
  using reflection
  by blast
qed
qed

```

```

lemma (in encoding) indRelR-reflects-pred-iff-indRelL-preserves-pred:
  fixes Pred :: ('procS, 'procT) Proc  $\Rightarrow$  bool
  shows rel-reflects-pred indRelR Pred = rel-preserves-pred indRelL Pred
proof

```

```

  assume reflection: rel-reflects-pred indRelR Pred
  show rel-preserves-pred indRelL Pred
proof clarify
  fix P Q

```

```

assume  $P \mathcal{R}[\cdot]L Q$ 
from this obtain  $S$  where  $S \in S Q$  and  $\llbracket S \rrbracket \in T P$ 
  by (induct, blast)
hence  $Q \mathcal{R}[\cdot]R P$ 
  by (simp add: indRelR.encR)
moreover assume  $Pred P$ 
ultimately show  $Pred Q$ 
  using reflection
  by blast
qed
next
assume preservation: rel-preserves-pred indRelL Pred
show rel-reflects-pred indRelR Pred
proof clarify
  fix  $P Q$ 
  assume  $P \mathcal{R}[\cdot]R Q$ 
  from this obtain  $S$  where  $S \in S P$  and  $\llbracket S \rrbracket \in T Q$ 
    by (induct, blast)
  hence  $Q \mathcal{R}[\cdot]L P$ 
    by (simp add: indRelL.encL)
  moreover assume  $Pred Q$ 
  ultimately show  $Pred P$ 
    using preservation
    by blast
qed
qed

lemma (in encoding) indRelR-reflects-binary-pred-iff-indRelL-preserves-binary-pred:
  fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$ 
  shows rel-reflects-binary-pred indRelR Pred = rel-preserves-binary-pred indRelL Pred
proof
  assume reflection: rel-reflects-binary-pred indRelR Pred
  show rel-preserves-binary-pred indRelL Pred
  proof clarify
    fix  $P Q x$ 
    assume  $P \mathcal{R}[\cdot]L Q$ 
    from this obtain  $S$  where  $S \in S Q$  and  $\llbracket S \rrbracket \in T P$ 
      by (induct, blast)
    hence  $Q \mathcal{R}[\cdot]R P$ 
      by (simp add: indRelR.encR)
    moreover assume  $Pred P x$ 
    ultimately show  $Pred Q x$ 
      using reflection
      by blast
    qed
  next
  assume preservation: rel-preserves-binary-pred indRelL Pred
  show rel-reflects-binary-pred indRelR Pred
  proof clarify
    fix  $P Q x$ 
    assume  $P \mathcal{R}[\cdot]R Q$ 
    from this obtain  $S$  where  $S \in S P$  and  $\llbracket S \rrbracket \in T Q$ 
      by (induct, blast)
    hence  $Q \mathcal{R}[\cdot]L P$ 
      by (simp add: indRelL.encL)
    moreover assume  $Pred Q x$ 
    ultimately show  $Pred P x$ 
      using preservation
      by blast
    qed
  qed

```

**lemma** (in *encoding*) *indRelR-respects-pred-iff-indRelL-respects-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $rel\text{-respects-pred } indRelR\ Pred = rel\text{-respects-pred } indRelL\ Pred$   
**using**  $indRelR\text{-preserves-pred-iff-indRelL-reflects-pred}$  [where  $Pred = Pred$ ]  
 $indRelR\text{-reflects-pred-iff-indRelL-preserves-pred}$  [where  $Pred = Pred$ ]  
**by** *blast*

**lemma** (in *encoding*) *indRelR-respects-binary-pred-iff-indRelL-respects-binary-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
**shows**  $rel\text{-respects-binary-pred } indRelR\ Pred = rel\text{-respects-binary-pred } indRelL\ Pred$   
**using**  $indRelR\text{-preserves-binary-pred-iff-indRelL-reflects-binary-pred}$  [where  $Pred = Pred$ ]  
 $indRelR\text{-reflects-binary-pred-iff-indRelL-preserves-binary-pred}$  [where  $Pred = Pred$ ]  
**by** *blast*

**lemma** (in *encoding*) *indRelR-cond-preservation-iff-indRelL-cond-reflection*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel\ Pred)$   
 $= (\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel\text{-reflects-pred } Rel\ Pred)$   
**proof**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel\ Pred$   
**then obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: rel\text{-preserves-pred } Rel\ Pred$   
**by** *blast*  
**from**  $A1$  **have**  $\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel^{-1}$   
**by** *simp*  
**moreover from**  $A2$  **have**  $rel\text{-reflects-pred } (Rel^{-1})\ Pred$   
**by** *simp*  
**ultimately show**  $\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel\text{-reflects-pred } Rel\ Pred$   
**by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel\text{-reflects-pred } Rel\ Pred$   
**then obtain**  $Rel$  **where**  $B1: \forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$   
**and**  $B2: rel\text{-reflects-pred } Rel\ Pred$   
**by** *blast*  
**from**  $B1$  **have**  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel^{-1}$   
**by** *simp*  
**moreover from**  $B2$  **have**  $rel\text{-preserves-pred } (Rel^{-1})\ Pred$   
**by** *blast*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel\ Pred$   
**by** *blast*  
**qed**

**lemma** (in *encoding*) *indRelR-cond-binary-preservation-iff-indRelL-cond-binary-reflection*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
**shows**  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel\ Pred)$   
 $= (\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge rel\text{-reflects-binary-pred } Rel\ Pred)$   
**proof**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel\ Pred$   
**then obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: rel\text{-preserves-binary-pred } Rel\ Pred$   
**by** *blast*  
**from**  $A1$  **have**  $\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel^{-1}$   
**by** *simp*  
**moreover from**  $A2$  **have**  $rel\text{-reflects-binary-pred } (Rel^{-1})\ Pred$   
**by** *simp*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel\ Pred$   
**by** *blast*

**next**  
**assume**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred}$   
**then obtain**  $Rel$  **where**  $B1: \forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$   
**and**  $B2: rel\text{-reflects-binary-pred } Rel \text{ Pred}$   
**by** *blast*  
**from**  $B1$  **have**  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel^{-1}$   
**by** *simp*  
**moreover from**  $B2$  **have**  $rel\text{-preserves-binary-pred } (Rel^{-1}) \text{ Pred}$   
**by** *simp*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ Pred}$   
**by** *blast*  
**qed**

**lemma** (in *encoding*) *indRelR-cond-reflection-iff-indRelL-cond-preservation:*

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$

**shows**  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ Pred})$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ Pred})$

**proof**

**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ Pred}$

**then obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: rel\text{-reflects-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $A1$  **have**  $\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel^{-1}$

**by** *simp*

**moreover from**  $A2$  **have**  $rel\text{-preserves-pred } (Rel^{-1}) \text{ Pred}$

**by** *blast*

**ultimately**

**show**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ Pred}$

**by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ Pred}$

**then obtain**  $Rel$  **where**  $B1: \forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$

**and**  $B2: rel\text{-preserves-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $B1$  **have**  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel^{-1}$

**by** *simp*

**moreover from**  $B2$  **have**  $rel\text{-reflects-pred } (Rel^{-1}) \text{ Pred}$

**by** *simp*

**ultimately**

**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ Pred}$

**by** *blast*

**qed**

**lemma** (in *encoding*) *indRelR-cond-binary-reflection-iff-indRelL-cond-binary-preservation:*

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$

**shows**  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred})$

$= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$   
 $\wedge rel\text{-preserves-binary-pred } Rel \text{ Pred})$

**proof**

**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred}$

**then obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: rel\text{-reflects-binary-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $A1$  **have**  $\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel^{-1}$

**by** *simp*

**moreover from**  $A2$  **have**  $rel\text{-preserves-binary-pred } (Rel^{-1}) \text{ Pred}$

**by** *blast*

**ultimately**

**show**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ Pred}$

**by** *blast*

**next**  
**assume**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ Pred}$   
**then obtain**  $Rel$  **where**  $B1: \forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$   
**and**  $B2: rel\text{-preserves-binary-pred } Rel \text{ Pred}$   
**by** *blast*  
**from**  $B1$  **have**  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel^{-1}$   
**by** *simp*  
**moreover from**  $B2$  **have**  $rel\text{-reflects-binary-pred } (Rel^{-1}) \text{ Pred}$   
**by** *simp*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred}$   
**by** *blast*  
**qed**

**lemma** (in *encoding*) *indRelR-cond-respection-iff-indRelL-cond-respection*:

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$

**shows**  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred})$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred})$

**proof**

**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred}$

**from this obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: rel\text{-respects-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $A1$  **have**  $\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in \{(a, b). (b, a) \in Rel\}$

**by** *simp*

**moreover from**  $A2$  **have**  $rel\text{-respects-pred } \{(a, b). (b, a) \in Rel\} \text{ Pred}$

**by** *blast*

**ultimately show**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred}$

**by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred}$

**from this obtain**  $Rel$  **where**  $A1: \forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$

**and**  $A2: rel\text{-respects-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $A1$  **have**  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in \{(a, b). (b, a) \in Rel\}$

**by** *simp*

**moreover from**  $A2$  **have**  $rel\text{-respects-pred } \{(a, b). (b, a) \in Rel\} \text{ Pred}$

**by** *blast*

**ultimately show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred}$

**by** *blast*

**qed**

**lemma** (in *encoding*) *indRelR-cond-binary-respection-iff-indRelL-cond-binary-respection*:

**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$

**shows**  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-binary-pred } Rel \text{ Pred})$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$   
 $\wedge rel\text{-respects-binary-pred } Rel \text{ Pred})$

**proof**

**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-binary-pred } Rel \text{ Pred}$

**from this obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: rel\text{-respects-binary-pred } Rel \text{ Pred}$

**by** *blast*

**from**  $A1$  **have**  $\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in \{(a, b). (b, a) \in Rel\}$

**by** *simp*

**moreover from**  $A2$  **have**  $rel\text{-respects-binary-pred } \{(a, b). (b, a) \in Rel\} \text{ Pred}$

**by** *blast*

**ultimately**

**show**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-binary-pred } Rel \text{ Pred}$

**by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-binary-pred } Rel \text{ Pred}$

**from this obtain**  $Rel$  **where**  $A1: \forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$   
**and**  $A2: rel-respects-binary-pred Rel Pred$   
**by** *blast*  
**from**  $A1$  **have**  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in \{(a, b). (b, a) \in Rel\}$   
**by** *simp*  
**moreover from**  $A2$  **have**  $rel-respects-binary-pred \{(a, b). (b, a) \in Rel\} Pred$   
**by** *blast*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-binary-pred Rel Pred$   
**by** *blast*  
**qed**

An encoding preserves, reflects, or respects a predicate iff  $indRelL$  reflects, preserves, or respects this predicate.

**lemma** (**in** *encoding*) *enc-preserves-pred-iff-indRelL-reflects-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-preserves-pred Pred = rel-reflects-pred indRelL Pred$   
**using**  $enc-preserves-pred-iff-indRelR-preserves-pred[where\ Pred=Pred]$   
 $indRelR-preserves-pred-iff-indRelL-reflects-pred[where\ Pred=Pred]$   
**by** *blast*

**lemma** (**in** *encoding*) *enc-reflects-pred-iff-indRelL-preserves-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-reflects-pred Pred = rel-preserves-pred indRelL Pred$   
**using**  $enc-reflects-pred-iff-indRelR-reflects-pred[where\ Pred=Pred]$   
 $indRelR-reflects-pred-iff-indRelL-preserves-pred[where\ Pred=Pred]$   
**by** *blast*

**lemma** (**in** *encoding*) *enc-respects-pred-iff-indRelL-respects-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-respects-pred Pred = rel-respects-pred indRelL Pred$   
**using**  $enc-preserves-pred-iff-indRelL-reflects-pred[where\ Pred=Pred]$   
 $enc-reflects-pred-iff-indRelL-preserves-pred[where\ Pred=Pred]$   
**by** *blast*

An encoding preserves, reflects, or respects a predicate iff there exists a relation, namely  $indRelL$ , that relates literal translations with their source terms and reflects, preserves, or respects this predicate.

**lemma** (**in** *encoding*) *enc-preserves-pred-iff-source-target-rel-reflects-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-preserves-pred Pred$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-reflects-pred Rel Pred)$   
**using**  $enc-preserves-pred-iff-source-target-rel-preserves-pred[where\ Pred=Pred]$   
 $indRelR-cond-preservation-iff-indRelL-cond-reflection[where\ Pred=Pred]$   
**by** *simp*

**lemma** (**in** *encoding*) *enc-reflects-pred-iff-source-target-rel-preserves-pred*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-reflects-pred Pred$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-preserves-pred Rel Pred)$   
**using**  $enc-reflects-pred-iff-source-target-rel-reflects-pred[where\ Pred=Pred]$   
 $indRelR-cond-reflection-iff-indRelL-cond-preservation[where\ Pred=Pred]$   
**by** *simp*

**lemma** (**in** *encoding*) *enc-respects-pred-iff-source-target-rel-respects-pred-encL*:  
**fixes**  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$   
**shows**  $enc-respects-pred Pred$   
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-respects-pred Rel Pred)$   
**using**  $enc-respects-pred-iff-source-target-rel-respects-pred-encR[where\ Pred=Pred]$   
 $indRelR-cond-respection-iff-indRelL-cond-respection[where\ Pred=Pred]$   
**by** *simp*

To analyse the respectation of source term behaviours we use relations that contain both kind of pairs:  $(S, \text{enc } S)$  as well as  $(\text{enc } S, S)$  for all source terms  $S$ .

**inductive-set** (in *encoding*) *indRel* ::  
 :: (((*'procS*, *'procT*) *Proc*) × ((*'procS*, *'procT*) *Proc*)) *set*  
**where**  
*encR*: (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *indRel* |  
*encL*: (*TargetTerm* ( $\llbracket S \rrbracket$ ), *SourceTerm S*) ∈ *indRel*

**abbreviation** (in *encoding*) *indRelInfix* ::  
 (*'procS*, *'procT*) *Proc* ⇒ (*'procS*, *'procT*) *Proc* ⇒ *bool* (-  $\mathcal{R}[\cdot]$  - [75, 75] 80)  
**where**  
*P*  $\mathcal{R}[\cdot]$  *Q* ≡ (*P*, *Q*) ∈ *indRel*

**lemma** (in *encoding*) *indRel-symm*:  
**shows** *sym indRel*  
**unfolding** *sym-def*  
**by** (*auto simp add: indRel.simps indRel.encR indRel.encL*)

**inductive-set** (in *encoding*) *indRelEQ* ::  
 :: (((*'procS*, *'procT*) *Proc*) × ((*'procS*, *'procT*) *Proc*)) *set*  
**where**  
*encR*: (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *indRelEQ* |  
*encL*: (*TargetTerm* ( $\llbracket S \rrbracket$ ), *SourceTerm S*) ∈ *indRelEQ* |  
*target*: (*TargetTerm T*, *TargetTerm T*) ∈ *indRelEQ* |  
*trans*:  $\llbracket (P, Q) \in \text{indRelEQ}; (Q, R) \in \text{indRelEQ} \rrbracket \implies (P, R) \in \text{indRelEQ}$

**abbreviation** (in *encoding*) *indRelEQinfix* ::  
 (*'procS*, *'procT*) *Proc* ⇒ (*'procS*, *'procT*) *Proc* ⇒ *bool* (-  $\sim[\cdot]$  - [75, 75] 80)  
**where**  
*P*  $\sim[\cdot]$  *Q* ≡ (*P*, *Q*) ∈ *indRelEQ*

**lemma** (in *encoding*) *indRelEQ-refl*:  
**shows** *refl indRelEQ*  
**unfolding** *refl-on-def*

**proof** *auto*  
**fix** *P*  
**show** *P*  $\sim[\cdot]$  *P*  
**proof** (*cases P*)  
**case** (*SourceTerm SP*)  
**assume** *SP* ∈ *S P*  
**moreover have** *SourceTerm SP*  $\sim[\cdot]$  *TargetTerm* ( $\llbracket SP \rrbracket$ )  
**by** (*rule indRelEQ.encR*)  
**moreover have** *TargetTerm* ( $\llbracket SP \rrbracket$ )  $\sim[\cdot]$  *SourceTerm SP*  
**by** (*rule indRelEQ.encL*)  
**ultimately show** *P*  $\sim[\cdot]$  *P*  
**by** (*simp add: indRelEQ.trans* [**where** *P*=*SourceTerm SP* **and** *Q*=*TargetTerm* ( $\llbracket SP \rrbracket$ )])  
**next**  
**case** (*TargetTerm TP*)  
**assume** *TP* ∈ *T P*  
**thus** *P*  $\sim[\cdot]$  *P*  
**by** (*simp add: indRelEQ.target*)  
**qed**  
**qed**

**lemma** (in *encoding*) *indRelEQ-is-preorder*:  
**shows** *preorder indRelEQ*  
**unfolding** *preorder-on-def*  
**proof**  
**show** *refl indRelEQ*  
**by** (*rule indRelEQ-refl*)





```

  case (target T)
  show (TargetTerm T, TargetTerm T) ∈ indRel*
  by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ indRel* and (Q, R) ∈ indRel*
  thus (P, R) ∈ indRel*
  by simp
qed
next
  fix P Q
  assume (P, Q) ∈ indRel*
  thus P ∼[·] Q
  proof induct
    show P ∼[·] P
    using indRelEQ-refl
    unfolding refl-on-def
    by simp
  next
    case (step Q R)
    assume P ∼[·] Q
    moreover assume Q  $\mathcal{R}$ [·] R
    hence Q ∼[·] R
    by (induct, simp-all add: indRelEQ.encR indRelEQ.encL)
  ultimately show P ∼[·] R
  by (rule indRelEQ.trans)
qed
qed

lemma (in encoding) refl-symm-trans-closure-of-indRel:
  shows indRelEQ = (symcl (indRel=))+
proof -
  have (symcl (indRel=))+ = (symcl indRel)*
  by (rule refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRel])
  moreover have symcl indRel = indRel
  by (simp add: indRel-symm symm-closure-of-symm-rel[where Rel=indRel])
  ultimately show indRelEQ = (symcl (indRel=))+
  by (simp add: refl-trans-closure-of-indRel)
qed

lemma (in encoding) symm-closure-of-indRelR:
  shows indRel = symcl indRelR
  and indRelEQ = (symcl (indRelR=))+
proof -
  show indRel = symcl indRelR
  proof auto
    fix P Q
    assume P  $\mathcal{R}$ [·] Q
    thus (P, Q) ∈ symcl indRelR
    by (induct, simp-all add: symcl-def indRelR.encR)
  next
    fix P Q
    assume (P, Q) ∈ symcl indRelR
    thus P  $\mathcal{R}$ [·] Q
    by (auto simp add: symcl-def indRelR.simps indRel.encR indRel.encL)
  qed
  thus indRelEQ = (symcl (indRelR=))+
  using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelR]
  refl-trans-closure-of-indRel
  by simp
qed

```

```

lemma (in encoding) symm-closure-of-indRelL:
  shows  $indRel = symcl\ indRelL$ 
  and  $indRelEQ = (symcl\ (indRelL^=))^+$ 
proof -
  show  $indRel = symcl\ indRelL$ 
  proof auto
  fix  $P\ Q$ 
  assume  $P\ \mathcal{R}[\cdot]\ Q$ 
  thus  $(P, Q) \in symcl\ indRelL$ 
  by (induct, simp-all add: symcl-def indRelL.encL)
next
  fix  $P\ Q$ 
  assume  $(P, Q) \in symcl\ indRelL$ 
  thus  $P\ \mathcal{R}[\cdot]\ Q$ 
  by (auto simp add: symcl-def indRelL.simps indRel.encR indRel.encL)
qed
thus  $indRelEQ = (symcl\ (indRelL^=))^+$ 
  using refl-symm-trans-closure-is-symm-refl-trans-closure [where  $Rel=indRelL$ ]
  refl-trans-closure-of-indRel
  by simp
qed

```

The relation  $indRel$  is a combination of  $indRelL$  and  $indRelR$ .  $indRel$  respects a predicate iff  $indRelR$  (or  $indRelL$ ) respects it.

```

lemma (in encoding) indRel-respects-pred-iff-indRelR-respects-pred:
  fixes  $Pred :: ('procS, 'procT)\ Proc \Rightarrow bool$ 
  shows  $rel-respects-pred\ indRel\ Pred = rel-respects-pred\ indRelR\ Pred$ 
proof
  assume respection: rel-respects-pred indRel Pred
  show  $rel-respects-pred\ indRelR\ Pred$ 
  proof auto
  fix  $P\ Q$ 
  assume  $P\ \mathcal{R}[\cdot]\ R\ Q$ 
  from this obtain  $S$  where  $S \in S\ P$  and  $\llbracket S \rrbracket \in T\ Q$ 
  by (induct, blast)
  hence  $P\ \mathcal{R}[\cdot]\ Q$ 
  by (simp add: indRel.encR)
  moreover assume  $Pred\ P$ 
  ultimately show  $Pred\ Q$ 
  using respection
  by blast
next
  fix  $P\ Q$ 
  assume  $P\ \mathcal{R}[\cdot]\ R\ Q$ 
  from this obtain  $S$  where  $S \in S\ P$  and  $\llbracket S \rrbracket \in T\ Q$ 
  by (induct, blast)
  hence  $P\ \mathcal{R}[\cdot]\ Q$ 
  by (simp add: indRel.encR)
  moreover assume  $Pred\ Q$ 
  ultimately show  $Pred\ P$ 
  using respection
  by blast
qed
next
  assume  $rel-respects-pred\ indRelR\ Pred$ 
  thus  $rel-respects-pred\ indRel\ Pred$ 
  using symm-closure-of-indRelR(1)
  respection-and-closures(2) [where  $Rel=indRelR$  and  $Pred=Pred$ ]
  by blast

```

qed

**lemma** (in *encoding*) *indRel-respects-binary-pred-iff-indRelR-respects-binary-pred*:  
fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$   
shows *rel-respects-binary-pred indRel Pred = rel-respects-binary-pred indRelR Pred*

**proof**

assume *respection: rel-respects-binary-pred indRel Pred*

show *rel-respects-binary-pred indRelR Pred*

**proof** *auto*

fix  $P Q x$

assume  $P \mathcal{R}[\cdot]R Q$

from *this* obtain  $S$  where  $S \in S P$  and  $\llbracket S \rrbracket \in T Q$

by (*induct, blast*)

hence  $P \mathcal{R}[\cdot] Q$

by (*simp add: indRel.encR*)

moreover assume  $Pred P x$

ultimately show  $Pred Q x$

using *respection*

by *blast*

**next**

fix  $P Q x$

assume  $P \mathcal{R}[\cdot]R Q$

from *this* obtain  $S$  where  $S \in S P$  and  $\llbracket S \rrbracket \in T Q$

by (*induct, blast*)

hence  $P \mathcal{R}[\cdot] Q$

by (*simp add: indRel.encR*)

moreover assume  $Pred Q x$

ultimately show  $Pred P x$

using *respection*

by *blast*

qed

**next**

assume *rel-respects-binary-pred indRelR Pred*

thus *rel-respects-binary-pred indRel Pred*

using *symm-closure-of-indRelR(1)*

*respection-of-binary-predicates-and-closures(2)[where Rel=indRelR and Pred=Pred]*

by *blast*

qed

**lemma** (in *encoding*) *indRel-cond-respection-iff-indRelR-cond-respection*:

fixes  $Pred :: ('procS, 'procT) Proc \Rightarrow bool$

shows  $(\exists Rel.$

$(\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$   
 $\wedge rel-respects-pred Rel Pred)$

$= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred Rel Pred)$

**proof**

assume  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

$\wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-respects-pred Rel Pred$

from *this* obtain  $Rel$

where  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$

and *rel-respects-pred Rel Pred*

by *blast*

thus  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred Rel Pred$

by *blast*

**next**

assume  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred Rel Pred$

from *this* obtain  $Rel$  where  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

and  $A2: rel-respects-pred Rel Pred$

by *blast*

from  $A1$  have  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in symcl Rel$

$\wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in symcl Rel$

**by** (*simp add: symcl-def*)  
**moreover from** *A2* **have** *rel-respects-pred (symcl Rel) Pred*  
**using** *respection-and-closures(2)[where Rel=Rel and Pred=Pred]*  
**by** *blast*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge rel-respects-pred\ Rel\ Pred$   
**by** *blast*  
**qed**

**lemma** (*in encoding*) *indRel-cond-binary-respection-iff-indRelR-cond-binary-respection:*  
**fixes** *Pred :: ('procS, 'procT) Proc  $\Rightarrow$  'b  $\Rightarrow$  bool*  
**shows**  $(\exists Rel.$   
 $(\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge rel-respects-binary-pred\ Rel\ Pred)$   
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-respects-binary-pred\ Rel\ Pred)$

**proof**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-respects-binary-pred\ Rel\ Pred$   
**from** *this* **obtain** *Rel*  
**where**  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$   
**and** *rel-respects-binary-pred Rel Pred*  
**by** *blast*  
**thus**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-binary-pred\ Rel\ Pred$   
**by** *blast*

**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-binary-pred\ Rel\ Pred$   
**from** *this* **obtain** *Rel* **where** *A1:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$*   
**and** *A2: rel-respects-binary-pred Rel Pred*  
**by** *blast*  
**from** *A1* **have**  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in symcl\ Rel$   
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in symcl\ Rel$   
**by** (*simp add: symcl-def*)  
**moreover from** *A2* **have** *rel-respects-binary-pred (symcl Rel) Pred*  
**using** *respection-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]*  
**by** *blast*  
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge rel-respects-binary-pred\ Rel\ Pred$   
**by** *blast*  
**qed**

An encoding respects a predicate iff indRel respects this predicate.

**lemma** (*in encoding*) *enc-respects-pred-iff-indRel-respects-pred:*  
**fixes** *Pred :: ('procS, 'procT) Proc  $\Rightarrow$  bool*  
**shows** *enc-respects-pred Pred = rel-respects-pred indRel Pred*  
**using** *enc-respects-pred-iff-indRelR-respects-pred[where Pred=Pred]*  
*indRel-respects-pred-iff-indRelR-respects-pred[where Pred=Pred]*  
**by** *simp*

An encoding respects a predicate iff there exists a relation, namely indRel, that relates source terms and their literal translations in both directions and respects this predicate.

**lemma** (*in encoding*) *enc-respects-pred-iff-source-target-rel-respects-pred-encRL:*  
**fixes** *Pred :: ('procS, 'procT) Proc  $\Rightarrow$  bool*  
**shows** *enc-respects-pred Pred*  
 $= (\exists Rel.$   
 $(\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge rel-respects-pred\ Rel\ Pred)$   
**using** *enc-respects-pred-iff-source-target-rel-respects-pred-encR[where Pred=Pred]*

*indRel-cond-respection-iff-indRelR-cond-respection*[**where** *Pred=Pred*]  
**by** *simp*

## 5.2 Relations Induced by the Encoding and a Relation on Target Terms

Some encodability like e.g. operational correspondence are defined w.r.t. a relation on target terms. To analyse such criteria we include the respective target term relation in the considered relation on the disjoint union of source and target terms.

**inductive-set** (in *encoding*) *indRelRT*

$:: ('procT \times 'procT) set \Rightarrow (((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set$   
**for** *TRel*  $:: ('procT \times 'procT) set$

**where**

*encR*:  $(SourceTerm S, TargetTerm ([S])) \in indRelRT TRel \mid$

*target*:  $(T1, T2) \in TRel \Longrightarrow (TargetTerm T1, TargetTerm T2) \in indRelRT TRel$

**abbreviation** (in *encoding*) *indRelRTinfix*

$:: ('procS, 'procT) Proc \Rightarrow ('procT \times 'procT) set \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$   
 $(- \mathcal{R}[\cdot]RT<-> - [75, 75, 75] 80)$

**where**

$P \mathcal{R}[\cdot]RT<TRel> Q \equiv (P, Q) \in indRelRT TRel$

**inductive-set** (in *encoding*) *indRelRTPO*

$:: ('procT \times 'procT) set \Rightarrow (((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set$   
**for** *TRel*  $:: ('procT \times 'procT) set$

**where**

*encR*:  $(SourceTerm S, TargetTerm ([S])) \in indRelRTPO TRel \mid$

*source*:  $(SourceTerm S, SourceTerm S) \in indRelRTPO TRel \mid$

*target*:  $(T1, T2) \in TRel \Longrightarrow (TargetTerm T1, TargetTerm T2) \in indRelRTPO TRel \mid$

*trans*:  $[(P, Q) \in indRelRTPO TRel; (Q, R) \in indRelRTPO TRel] \Longrightarrow (P, R) \in indRelRTPO TRel$

**abbreviation** (in *encoding*) *indRelRTPOinfix*

$:: ('procS, 'procT) Proc \Rightarrow ('procT \times 'procT) set \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$   
 $(- \lesssim[\cdot]RT<-> - [75, 75, 75] 80)$

**where**

$P \lesssim[\cdot]RT<TRel> Q \equiv (P, Q) \in indRelRTPO TRel$

**lemma** (in *encoding*) *indRelRTPO-refl*:

**fixes** *TRel*  $:: ('procT \times 'procT) set$

**assumes** *refl*: *refl TRel*

**shows** *refl* (*indRelRTPO TRel*)

**unfolding** *refl-on-def*

**proof** *auto*

**fix** *P*

**show**  $P \lesssim[\cdot]RT<TRel> P$

**proof** (*cases P*)

**case** (*SourceTerm SP*)

**assume**  $SP \in S P$

**thus**  $P \lesssim[\cdot]RT<TRel> P$

**by** (*simp add: indRelRTPO.source*)

**next**

**case** (*TargetTerm TP*)

**assume**  $TP \in T P$

**with** *refl* **show**  $P \lesssim[\cdot]RT<TRel> P$

**unfolding** *refl-on-def*

**by** (*simp add: indRelRTPO.target*)

**qed**

**qed**

**lemma** (in *encoding*) *refl-trans-closure-of-indRelRT*:

**fixes** *TRel*  $:: ('procT \times 'procT) set$

```

assumes refl: refl TRel
shows indRelRTPO TRel = (indRelRT TRel)*
proof auto
  fix P Q
  assume P  $\lesssim$  [·]RT<TRel> Q
  thus (P, Q) ∈ (indRelRT TRel)*
  proof induct
    case (encR S)
    show (SourceTerm S, TargetTerm ([S])) ∈ (indRelRT TRel)*
      using indRelRT.encR[of S TRel]
      by simp
  next
    case (source S)
    show (SourceTerm S, SourceTerm S) ∈ (indRelRT TRel)*
      by simp
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    thus (TargetTerm T1, TargetTerm T2) ∈ (indRelRT TRel)*
      using indRelRT.target[of T1 T2 TRel]
      by simp
  next
    case (trans P Q R)
    assume (P, Q) ∈ (indRelRT TRel)* and (Q, R) ∈ (indRelRT TRel)*
    thus (P, R) ∈ (indRelRT TRel)*
      by simp
  qed
next
  fix P Q
  assume (P, Q) ∈ (indRelRT TRel)*
  thus P  $\lesssim$  [·]RT<TRel> Q
  proof induct
    from refl show P  $\lesssim$  [·]RT<TRel> P
      using indRelRTPO-refl[of TRel]
      unfolding refl-on-def
      by simp
  next
    case (step Q R)
    assume P  $\lesssim$  [·]RT<TRel> Q
    moreover assume Q  $\mathcal{R}$ [·]RT<TRel> R
    hence Q  $\lesssim$  [·]RT<TRel> R
      by (induct, simp-all add: indRelRTPO.encR indRelRTPO.target)
    ultimately show P  $\lesssim$  [·]RT<TRel> R
      by (rule indRelRTPO.trans)
  qed
qed

lemma (in encoding) indRelRTPO-is-preorder:
  fixes TRel :: ('procT × 'procT) set
  assumes reflT: refl TRel
  shows preorder (indRelRTPO TRel)
    unfolding preorder-on-def
proof
  from reflT show refl (indRelRTPO TRel)
    by (rule indRelRTPO-refl)
next
  show trans (indRelRTPO TRel)
    unfolding trans-def
proof clarify
  fix P Q R
  assume P  $\lesssim$  [·]RT<TRel> Q and Q  $\lesssim$  [·]RT<TRel> R

```

```

thus  $P \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} R$ 
  using indRelRTPO.trans
  by blast
qed
qed

```

```

lemma (in encoding) transitive-closure-of-TRel-to-indRelRTPO:
  fixes  $TRel :: ('procT \times 'procT) \text{ set}$ 
  and  $TP\ TQ :: 'procT$ 
  shows  $(TP, TQ) \in TRel^+ \implies \text{TargetTerm } TP \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TQ$ 
proof –
  assume  $(TP, TQ) \in TRel^+$ 
  thus  $\text{TargetTerm } TP \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TQ$ 
  proof induct
    fix  $TQ$ 
    assume  $(TP, TQ) \in TRel$ 
    thus  $\text{TargetTerm } TP \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TQ$ 
    by (rule indRelRTPO.target)
  next
    case (step TQ TR)
    assume  $\text{TargetTerm } TP \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TQ$ 
    moreover assume  $(TQ, TR) \in TRel$ 
    hence  $\text{TargetTerm } TQ \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TR$ 
    by (simp add: indRelRTPO.target)
    ultimately show  $\text{TargetTerm } TP \lesssim_{\llbracket \cdot \rrbracket RT < TRel >} \text{TargetTerm } TR$ 
    by (rule indRelRTPO.trans)
  qed
qed

```

The relation  $\text{indRelRT}$  is the smallest relation that relates all source terms and their literal translations and contains  $TRel$ . Thus there exists a relation that relates source terms and their literal translations and satisfies some predicate on its pairs iff the predicate holds for the pairs of  $\text{indRelR}$ .

```

lemma (in encoding) indRelR-modulo-pred-impl-indRelRT-modulo-pred:
  fixes  $Pred :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \implies \text{bool}$ 
  shows  $(\forall (P, Q) \in \text{indRelR}. Pred (P, Q)) = (\forall TRel. (\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ))) \longleftrightarrow (\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q))$ 
proof (rule iffI)
  assume  $A: \forall (P, Q) \in \text{indRelR}. Pred (P, Q)$ 
  show  $\forall TRel. (\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ)) = (\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q))$ 
  proof (rule allI, rule iffI)
    fix  $TRel$ 
    assume  $\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ)$ 
    with  $A$  show  $\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q)$ 
    by (auto simp add: indRelR.encR indRelRT.simps)
  next
    fix  $TRel$ 
    assume  $\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q)$ 
    thus  $\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ)$ 
    by (auto simp add: indRelRT.target)
  qed
next
  assume  $\forall TRel. (\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ)) \longleftrightarrow (\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q))$ 
  hence  $B: \bigwedge TRel. (\forall (TP, TQ) \in TRel. Pred (\text{TargetTerm } TP, \text{TargetTerm } TQ)) \longleftrightarrow (\forall (P, Q) \in \text{indRelRT } TRel. Pred (P, Q))$ 
  by blast
  have  $\bigwedge S. Pred (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))$ 
  using  $B[\text{of } \{\}]$ 
  by (simp add: indRelRT.simps)

```

**thus**  $\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q)$   
**by** (*auto simp add: indRelR.simps*)  
**qed**

**lemma** (*in encoding*) *indRelRT-iff-exists-source-target-relation*:  
**fixes**  $\text{Pred} :: (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \Rightarrow \text{bool}$   
**shows**  $(\forall TRel. (\forall (TP, TQ) \in TRel. \text{Pred } (\text{TargetTerm } TP, \text{TargetTerm } TQ)))$   
 $\iff (\forall (P, Q) \in \text{indRelRT } TRel. \text{Pred } (P, Q))$   
 $= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel) \wedge (\forall (P, Q) \in Rel. \text{Pred } (P, Q)))$   
**using** *indRelR-iff-exists-source-target-relation*[**where**  $\text{Pred} = \text{Pred}$ ]  
*indRelR-modulo-pred-impl-indRelRT-modulo-pred*[**where**  $\text{Pred} = \text{Pred}$ ]  
**by** *simp*

**lemma** (*in encoding*) *indRelRT-modulo-pred-impl-indRelRTPO-modulo-pred*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{set}$   
**and**  $\text{Pred} :: (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \Rightarrow \text{bool}$   
**assumes** *reflCond*:  $\forall P. \text{Pred } (P, P)$   
**and** *transCond*:  $\forall P Q R. \text{Pred } (P, Q) \wedge \text{Pred } (Q, R) \longrightarrow \text{Pred } (P, R)$   
**shows**  $(\forall (P, Q) \in \text{indRelRT } TRel. \text{Pred } (P, Q)) = (\forall (P, Q) \in \text{indRelRTPO } TRel. \text{Pred } (P, Q))$

**proof** *auto*

**fix**  $P Q$   
**assume**  $A: \forall x \in \text{indRelRT } TRel. \text{Pred } x$   
**assume**  $P \lesssim [\cdot]_{RT} < TRel > Q$   
**thus**  $\text{Pred } (P, Q)$   
**proof** *induct*  
**case** (*encR S*)  
**have**  $\text{SourceTerm } S \mathcal{R} [\cdot]_{RT} < TRel > \text{TargetTerm } (\llbracket S \rrbracket)$   
**by** (*simp add: indRelRT.encR*)  
**with**  $A$  **show**  $\text{Pred } (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket))$   
**by** *simp*  
**next**  
**case** (*source S*)  
**from** *reflCond* **show**  $\text{Pred } (\text{SourceTerm } S, \text{SourceTerm } S)$   
**by** *simp*  
**next**  
**case** (*target T1 T2*)  
**assume**  $(T1, T2) \in TRel$   
**hence**  $\text{TargetTerm } T1 \mathcal{R} [\cdot]_{RT} < TRel > \text{TargetTerm } T2$   
**by** (*simp add: indRelRT.target*)  
**with**  $A$  **show**  $\text{Pred } (\text{TargetTerm } T1, \text{TargetTerm } T2)$   
**by** *simp*  
**next**  
**case** (*trans P Q R*)  
**assume**  $\text{Pred } (P, Q)$  **and**  $\text{Pred } (Q, R)$   
**with** *transCond* **show**  $\text{Pred } (P, R)$   
**by** *blast*

**qed**

**next**  
**fix**  $P Q$   
**assume**  $\forall x \in \text{indRelRTPO } TRel. \text{Pred } x$  **and**  $P \mathcal{R} [\cdot]_{RT} < TRel > Q$   
**thus**  $\text{Pred } (P, Q)$   
**by** (*auto simp add: indRelRTPO.encR indRelRTPO.target indRelRT.simps*)  
**qed**

**lemma** (*in encoding*) *indRelR-modulo-pred-impl-indRelRTPO-modulo-pred*:  
**fixes**  $\text{Pred} :: (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \Rightarrow \text{bool}$   
**assumes**  $\forall P. \text{Pred } (P, P)$   
**and**  $\forall P Q R. \text{Pred } (P, Q) \wedge \text{Pred } (Q, R) \longrightarrow \text{Pred } (P, R)$   
**shows**  $(\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q))$   
 $= (\forall TRel. (\forall (TP, TQ) \in TRel. \text{Pred } (\text{TargetTerm } TP, \text{TargetTerm } TQ)))$   
 $\iff (\forall (P, Q) \in \text{indRelRTPO } TRel. \text{Pred } (P, Q))$



```

proof –
  have  $(\forall (P, Q) \in \text{indRelR}. \text{Pred} (P, Q)) = (\forall T\text{Rel}. (\forall (TP, TQ) \in T\text{Rel}. \text{Pred} (\text{TargetTerm } TP, \text{TargetTerm } TQ))) \longleftrightarrow (\forall (P, Q) \in \text{indRelRT } T\text{Rel}. \text{Pred} (P, Q))$ 
    using indRelR-modulo-pred-impl-indRelRT-modulo-pred [where Pred=Pred]
    by simp
  moreover
  have  $\forall T\text{Rel}. (\forall (P, Q) \in \text{indRelRT } T\text{Rel}. \text{Pred} (P, Q)) = (\forall (P, Q) \in \text{indRelRTPO } T\text{Rel}. \text{Pred} (P, Q))$ 
    using assms indRelRT-modulo-pred-impl-indRelRTPO-modulo-pred [where Pred=Pred]
    by blast
  ultimately show ?thesis
    by simp
qed

```

The relation *indRelLT* includes *TRel* and relates literal translations and their source terms.

```

inductive-set (in encoding) indRelLT
  :: ('procT × 'procT) set ⇒ (((('procS, 'procT) Proc) × ((('procS, 'procT) Proc))) set
  for TRel :: ('procT × 'procT) set
  where
  encL:  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{indRelLT } T\text{Rel} \mid$ 
  target:  $(T1, T2) \in T\text{Rel} \implies (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelLT } T\text{Rel}$ 

```

```

abbreviation (in encoding) indRelLTinfix
  :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
  (-  $\mathcal{R}[\cdot]LT<->$  - [75, 75, 75] 80)
  where
   $P \mathcal{R}[\cdot]LT<T\text{Rel}> Q \equiv (P, Q) \in \text{indRelLT } T\text{Rel}$ 

```

```

inductive-set (in encoding) indRelLTPO
  :: ('procT × 'procT) set ⇒ (((('procS, 'procT) Proc) × ((('procS, 'procT) Proc))) set
  for TRel :: ('procT × 'procT) set
  where
  encL:  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{indRelLTPO } T\text{Rel} \mid$ 
  source:  $(\text{SourceTerm } S, \text{SourceTerm } S) \in \text{indRelLTPO } T\text{Rel} \mid$ 
  target:  $(T1, T2) \in T\text{Rel} \implies (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelLTPO } T\text{Rel} \mid$ 
  trans:  $\llbracket (P, Q) \in \text{indRelLTPO } T\text{Rel}; (Q, R) \in \text{indRelLTPO } T\text{Rel} \rrbracket \implies (P, R) \in \text{indRelLTPO } T\text{Rel}$ 

```

```

abbreviation (in encoding) indRelLTPOinfix
  :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
  (-  $\lesssim[\cdot]LT<->$  - [75, 75, 75] 80)
  where
   $P \lesssim[\cdot]LT<T\text{Rel}> Q \equiv (P, Q) \in \text{indRelLTPO } T\text{Rel}$ 

```

**lemma** (**in encoding**) *indRelLTPO-refl*:

```

fixes TRel :: ('procT × 'procT) set
assumes refl: refl TRel
shows refl (indRelLTPO TRel)
  unfolding refl-on-def

```

**proof** *auto*

```

fix P
show  $P \lesssim[\cdot]LT<T\text{Rel}> P$ 
proof (cases P)
  case (SourceTerm SP)
  assume  $SP \in S \ P$ 
  thus  $P \lesssim[\cdot]LT<T\text{Rel}> P$ 
    by (simp add: indRelLTPO.source)

```

**next**

```

case (TargetTerm TP)
assume  $TP \in T \ P$ 
with refl show  $P \lesssim[\cdot]LT<T\text{Rel}> P$ 
  using indRelLTPO.target [of TP TP TRel]
  unfolding refl-on-def

```

by *simp*  
qed  
qed

**lemma** (in *encoding*) *refl-trans-closure-of-indRelLT*:

fixes  $TRel :: ('procT \times 'procT) set$

assumes *refl*:  $refl\ TRel$

shows  $indRelLTPO\ TRel = (indRelLT\ TRel)^*$

**proof** *auto*

fix  $P\ Q$

assume  $P \lesssim [\cdot]LT < TRel > Q$

thus  $(P, Q) \in (indRelLT\ TRel)^*$

**proof** *induct*

case (*encL*  $S$ )

show  $(TargetTerm\ ([S]), SourceTerm\ S) \in (indRelLT\ TRel)^*$

using  $indRelLT.encL[of\ S\ TRel]$

by *simp*

**next**

case (*source*  $S$ )

show  $(SourceTerm\ S, SourceTerm\ S) \in (indRelLT\ TRel)^*$

by *simp*

**next**

case (*target*  $T1\ T2$ )

assume  $(T1, T2) \in TRel$

thus  $(TargetTerm\ T1, TargetTerm\ T2) \in (indRelLT\ TRel)^*$

using  $indRelLT.target[of\ T1\ T2\ TRel]$

by *simp*

**next**

case (*trans*  $P\ Q\ R$ )

assume  $(P, Q) \in (indRelLT\ TRel)^*$  and  $(Q, R) \in (indRelLT\ TRel)^*$

thus  $(P, R) \in (indRelLT\ TRel)^*$

by *simp*

qed

**next**

fix  $P\ Q$

assume  $(P, Q) \in (indRelLT\ TRel)^*$

thus  $P \lesssim [\cdot]LT < TRel > Q$

**proof** *induct*

from *refl* show  $P \lesssim [\cdot]LT < TRel > P$

using  $indRelLTPO-refl[of\ TRel]$

unfolding *refl-on-def*

by *simp*

**next**

case (*step*  $Q\ R$ )

assume  $P \lesssim [\cdot]LT < TRel > Q$

moreover assume  $Q \mathcal{R} [\cdot]LT < TRel > R$

hence  $Q \lesssim [\cdot]LT < TRel > R$

by (*induct*, *simp-all* add:  $indRelLTPO.encL\ indRelLTPO.target$ )

ultimately show  $P \lesssim [\cdot]LT < TRel > R$

by (*rule*  $indRelLTPO.trans$ )

qed

qed

**inductive-set** (in *encoding*) *indRelT*

$:: ('procT \times 'procT) set \Rightarrow (((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set$

for  $TRel :: ('procT \times 'procT) set$

where

*encR*:  $(SourceTerm\ S, TargetTerm\ ([S])) \in indRelT\ TRel \mid$

*encL*:  $(TargetTerm\ ([S]), SourceTerm\ S) \in indRelT\ TRel \mid$

*target*:  $(T1, T2) \in TRel \implies (TargetTerm\ T1, TargetTerm\ T2) \in indRelT\ TRel$

**abbreviation** (in *encoding*) *indRelT*infix

$:: ('procS, 'procT) Proc \Rightarrow ('procT \times 'procT) set \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$   
 $(- \mathcal{R}[\cdot] T <-> - [75, 75, 75] 80)$

**where**

$P \mathcal{R}[\cdot] T < TRel > Q \equiv (P, Q) \in indRelT TRel$

**lemma** (in *encoding*) *indRelT-symm*:

**fixes** *TRel*  $:: ('procT \times 'procT) set$

**assumes** *symm*: *sym TRel*

**shows** *sym* (*indRelT TRel*)

**unfolding** *sym-def*

**proof** *clarify*

**fix** *P Q*

**assume**  $(P, Q) \in indRelT TRel$

**thus**  $(Q, P) \in indRelT TRel$

**using** *symm*

**unfolding** *sym-def*

**by** (*induct*, *simp-all add: indRelT.encL indRelT.encR indRelT.target*)

**qed**

**inductive-set** (in *encoding*) *indRelTEQ*

$:: ('procT \times 'procT) set \Rightarrow (((('procS, 'procT) Proc) \times ((('procS, 'procT) Proc))) set$

**for** *TRel*  $:: ('procT \times 'procT) set$

**where**

*encR*:  $(SourceTerm S, TargetTerm ([S])) \in indRelTEQ TRel \mid$

*encL*:  $(TargetTerm ([S]), SourceTerm S) \in indRelTEQ TRel \mid$

*target*:  $(T1, T2) \in TRel \implies (TargetTerm T1, TargetTerm T2) \in indRelTEQ TRel \mid$

*trans*:  $[(P, Q) \in indRelTEQ TRel; (Q, R) \in indRelTEQ TRel] \implies (P, R) \in indRelTEQ TRel$

**abbreviation** (in *encoding*) *indRelTEQ*infix

$:: ('procS, 'procT) Proc \Rightarrow ('procT \times 'procT) set \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$   
 $(- \sim[\cdot] T <-> - [75, 75, 75] 80)$

**where**

$P \sim[\cdot] T < TRel > Q \equiv (P, Q) \in indRelTEQ TRel$

**lemma** (in *encoding*) *indRelTEQ-refl*:

**fixes** *TRel*  $:: ('procT \times 'procT) set$

**assumes** *refl*: *refl TRel*

**shows** *refl* (*indRelTEQ TRel*)

**unfolding** *refl-on-def*

**proof** *auto*

**fix** *P*

**show**  $P \sim[\cdot] T < TRel > P$

**proof** (*cases P*)

**case** (*SourceTerm SP*)

**assume**  $SP \in S P$

**moreover have** *SourceTerm SP*  $\sim[\cdot] T < TRel > TargetTerm ([SP])$

**by** (*rule indRelTEQ.encR*)

**moreover have** *TargetTerm ([SP])*  $\sim[\cdot] T < TRel > SourceTerm SP$

**by** (*rule indRelTEQ.encL*)

**ultimately show**  $P \sim[\cdot] T < TRel > P$

**by** (*simp add: indRelTEQ.trans* [**where**  $P = SourceTerm SP$  **and**  $Q = TargetTerm ([SP])$ ])

**next**

**case** (*TargetTerm TP*)

**assume**  $TP \in T P$

**with** *refl* **show**  $P \sim[\cdot] T < TRel > P$

**unfolding** *refl-on-def*

**by** (*simp add: indRelTEQ.target*)

**qed**

**qed**

```

lemma (in encoding) indRelTEQ-symm:
  fixes TRel :: ('procT × 'procT) set
  assumes symm: sym TRel
  shows sym (indRelTEQ TRel)
    unfolding sym-def
proof clarify
  fix P Q
  assume  $P \sim[\cdot] T < TRel > Q$ 
  thus  $Q \sim[\cdot] T < TRel > P$ 
  proof induct
    case (encR S)
    show TargetTerm ( $\llbracket S \rrbracket$ )  $\sim[\cdot] T < TRel >$  SourceTerm S
      by (rule indRelTEQ.encL)
    next
    case (encL S)
    show SourceTerm S  $\sim[\cdot] T < TRel >$  TargetTerm ( $\llbracket S \rrbracket$ )
      by (rule indRelTEQ.encR)
    next
    case (target T1 T2)
    assume (T1, T2)  $\in$  TRel
    with symm show TargetTerm T2  $\sim[\cdot] T < TRel >$  TargetTerm T1
      unfolding sym-def
      by (simp add: indRelTEQ.target)
    next
    case (trans P Q R)
    assume  $R \sim[\cdot] T < TRel > Q$  and  $Q \sim[\cdot] T < TRel > P$ 
    thus  $R \sim[\cdot] T < TRel > P$ 
      by (rule indRelTEQ.trans)
  qed
qed

```

```

lemma (in encoding) refl-trans-closure-of-indRelT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows indRelTEQ TRel = (indRelT TRel)*
proof auto
  fix P Q
  assume  $P \sim[\cdot] T < TRel > Q$ 
  thus (P, Q)  $\in$  (indRelT TRel)*
  proof induct
    case (encR S)
    show (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  (indRelT TRel)*
      using indRelT.encR[of S TRel]
      by simp
    next
    case (encL S)
    show (TargetTerm ( $\llbracket S \rrbracket$ ), SourceTerm S)  $\in$  (indRelT TRel)*
      using indRelT.encL[of S TRel]
      by simp
    next
    case (target T1 T2)
    assume (T1, T2)  $\in$  TRel
    thus (TargetTerm T1, TargetTerm T2)  $\in$  (indRelT TRel)*
      using indRelT.target[of T1 T2 TRel]
      by simp
    next
    case (trans P Q R)
    assume (P, Q)  $\in$  (indRelT TRel)* and (Q, R)  $\in$  (indRelT TRel)*
    thus (P, R)  $\in$  (indRelT TRel)*
      by simp
  qed

```

```

next
  fix P Q
  assume (P, Q) ∈ (indRelT TRel)*
  thus P ~[·]T<TRel> Q
  proof induct
    from refl show P ~[·]T<TRel> P
      using indRelTEQ-refl[of TRel]
      unfolding refl-on-def
      by simp
  next
  case (step Q R)
  assume P ~[·]T<TRel> Q
  moreover assume Q R[·]T<TRel> R
  hence Q ~[·]T<TRel> R
    by (induct, simp-all add: indRelTEQ.encR indRelTEQ.encL indRelTEQ.target)
  ultimately show P ~[·]T<TRel> R
    by (rule indRelTEQ.trans)
qed

```

```

lemma (in encoding) refl-symm-trans-closure-of-indRelT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
    and symm: sym TRel
  shows indRelTEQ TRel = (symcl ((indRelT TRel)=))+
proof -
  have (symcl ((indRelT TRel)=))+ = (symcl (indRelT TRel))*
    by (rule refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelT TRel])
  moreover from symm have symcl (indRelT TRel) = indRelT TRel
    using indRelT-symm[where TRel=TRel] symm-closure-of-symm-rel[where Rel=indRelT TRel]
    by blast
  ultimately show indRelTEQ TRel = (symcl ((indRelT TRel)=))+
    using refl refl-trans-closure-of-indRelT[where TRel=TRel]
    by simp
qed

```

```

lemma (in encoding) symm-closure-of-indRelRT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
    and symm: sym TRel
  shows indRelT TRel = symcl (indRelRT TRel)
    and indRelTEQ TRel = (symcl ((indRelRT TRel)=))+
proof -
  show indRelT TRel = symcl (indRelRT TRel)
  proof auto
    fix P Q
    assume P R[·]T<TRel> Q
    thus (P, Q) ∈ symcl (indRelRT TRel)
      by (induct, simp-all add: symcl-def indRelRT.encR indRelRT.target)
  next
  fix P Q
  assume (P, Q) ∈ symcl (indRelRT TRel)
  thus P R[·]T<TRel> Q
  proof (auto simp add: symcl-def indRelRT.simps)
    fix S
    show SourceTerm S R[·]T<TRel> TargetTerm ([S])
      by (rule indRelT.encR)
  next
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1 R[·]T<TRel> TargetTerm T2

```

```

    by (rule indRelT.target)
next
fix S
show TargetTerm ([[S]])  $\mathcal{R}[\cdot]T < TRel >$  SourceTerm S
  by (rule indRelT.encL)
next
fix T1 T2
assume (T1, T2)  $\in TRel$ 
with symm show TargetTerm T2  $\mathcal{R}[\cdot]T < TRel >$  TargetTerm T1
  unfolding sym-def
  by (simp add: indRelT.target)
qed
qed
with refl show indRelTEQ TRel = (symcl ((indRelRT TRel)=))+
  using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelRT TRel]
  refl-trans-closure-of-indRelT
  by simp
qed

lemma (in encoding) symm-closure-of-indRelLT:
fixes TRel :: ('procT  $\times$  'procT) set
assumes refl: refl TRel
  and symm: sym TRel
shows indRelT TRel = symcl (indRelLT TRel)
  and indRelTEQ TRel = (symcl ((indRelLT TRel)=))+
proof -
show indRelT TRel = symcl (indRelLT TRel)
proof auto
fix P Q
assume P  $\mathcal{R}[\cdot]T < TRel >$  Q
thus (P, Q)  $\in$  symcl (indRelLT TRel)
  by (induct, simp-all add: symcl-def indRelLT.encL indRelLT.target)
next
fix P Q
assume (P, Q)  $\in$  symcl (indRelLT TRel)
thus P  $\mathcal{R}[\cdot]T < TRel >$  Q
proof (auto simp add: symcl-def indRelLT.simps)
fix S
show SourceTerm S  $\mathcal{R}[\cdot]T < TRel >$  TargetTerm ([[S]])
  by (rule indRelT.encR)
next
fix T1 T2
assume (T1, T2)  $\in TRel$ 
thus TargetTerm T1  $\mathcal{R}[\cdot]T < TRel >$  TargetTerm T2
  by (rule indRelT.target)
next
fix S
show TargetTerm ([[S]])  $\mathcal{R}[\cdot]T < TRel >$  SourceTerm S
  by (rule indRelT.encL)
next
fix T1 T2
assume (T1, T2)  $\in TRel$ 
with symm show TargetTerm T2  $\mathcal{R}[\cdot]T < TRel >$  TargetTerm T1
  unfolding sym-def
  by (simp add: indRelT.target)
qed
qed
with refl show indRelTEQ TRel = (symcl ((indRelLT TRel)=))+
  using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelLT TRel]
  refl-trans-closure-of-indRelT
  by simp

```

qed

If the relations  $\text{indRelRT}$ ,  $\text{indRelLT}$ , or  $\text{indRelT}$  contain a pair of target terms, then this pair is also related by the considered target term relation.

**lemma** (*in encoding*) *indRelRT-to-TRel*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $TP\ TQ :: 'procT$   
**assumes**  $rel: \text{TargetTerm } TP\ \mathcal{R}[\cdot]RT < TRel > \text{TargetTerm } TQ$   
**shows**  $(TP, TQ) \in TRel$   
**using**  $rel$   
**by** (*simp add: indRelRT.simps*)

**lemma** (*in encoding*) *indRelLT-to-TRel*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $TP\ TQ :: 'procT$   
**assumes**  $rel: \text{TargetTerm } TP\ \mathcal{R}[\cdot]LT < TRel > \text{TargetTerm } TQ$   
**shows**  $(TP, TQ) \in TRel$   
**using**  $rel$   
**by** (*simp add: indRelLT.simps*)

**lemma** (*in encoding*) *indRelT-to-TRel*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $TP\ TQ :: 'procT$   
**assumes**  $rel: \text{TargetTerm } TP\ \mathcal{R}[\cdot]T < TRel > \text{TargetTerm } TQ$   
**shows**  $(TP, TQ) \in TRel$   
**using**  $rel$   
**by** (*simp add: indRelT.simps*)

If the preorders  $\text{indRelRTPO}$ ,  $\text{indRelLTPO}$ , or the equivalence  $\text{indRelTEQ}$  contain a pair of terms, then the pair of target terms that is related to these two terms is also related by the reflexive and transitive closure of the considered target term relation.

**lemma** (*in encoding*) *indRelRTPO-to-TRel*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $P\ Q :: ('procS, 'procT) \text{ Proc}$   
**assumes**  $rel: P \lesssim [\cdot]RT < TRel > Q$   
**shows**  $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow SP = SQ$   
**and**  $\forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q$   
 $\longrightarrow ([SP], TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \wedge T2 = [S]\})^+$   
**and**  $\forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow \text{False}$   
**and**  $\forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$   
**proof** –  
**have**  $reflTRel: \forall S. ([S], [S]) \in TRel \cup \{(T1, T2). \exists S. T1 = [S] \wedge T2 = [S]\}$   
**by** *auto*  
**from**  $rel$  **show**  $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow SP = SQ$   
**and**  $\forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q$   
 $\longrightarrow ([SP], TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \wedge T2 = [S]\})^+$   
**and**  $\forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow \text{False}$   
**and**  $\forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$   
**proof** *induct*  
**case** (*encR S*)  
**show**  $\forall SP\ SQ. SP \in S\ \text{SourceTerm } S \wedge SQ \in S\ \text{TargetTerm } ([S]) \longrightarrow SP = SQ$   
**and**  $\forall TP\ SQ. TP \in T\ \text{SourceTerm } S \wedge SQ \in S\ \text{TargetTerm } ([S]) \longrightarrow \text{False}$   
**and**  $\forall TP\ TQ. TP \in T\ \text{SourceTerm } S \wedge TQ \in T\ \text{TargetTerm } ([S]) \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp-all*  
**from**  $reflTRel$  **show**  $\forall SP\ TQ. SP \in S\ \text{SourceTerm } S \wedge TQ \in T\ \text{TargetTerm } ([S])$   
 $\longrightarrow ([SP], TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \wedge T2 = [S]\})^+$   
**by** *blast*  
**next**  
**case** (*source S*)  
**show**  $\forall SP\ SQ. SP \in S\ \text{SourceTerm } S \wedge SQ \in S\ \text{SourceTerm } S \longrightarrow SP = SQ$

by *simp*  
**show**  $\forall SP\ TQ. SP \in S\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP\ SQ. TP \in T\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S \longrightarrow False$   
**and**  $\forall TP\ TQ. TP \in T\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S \longrightarrow (TP, TQ) \in TRel^+$   
 by *simp-all*  
**next**  
**case** (*target T1 T2*)  
**show**  $\forall SP\ SQ. SP \in S\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2 \longrightarrow SP = SQ$   
**and**  $\forall SP\ TQ. SP \in S\ TargetTerm\ T1 \wedge TQ \in T\ TargetTerm\ T2$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP\ SQ. TP \in T\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2 \longrightarrow False$   
 by *simp-all*  
**assume**  $(T1, T2) \in TRel$   
**thus**  $\forall TP\ TQ. TP \in T\ TargetTerm\ T1 \wedge TQ \in T\ TargetTerm\ T2 \longrightarrow (TP, TQ) \in TRel^+$   
 by *simp*  
**next**  
**case** (*trans P Q R*)  
**assume**  $A1: \forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow SP = SQ$   
**and**  $A2: \forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A3: \forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow False$   
**and**  $A4: \forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$   
**and**  $A5: \forall SQ\ SR. SQ \in S\ Q \wedge SR \in S\ R \longrightarrow SQ = SR$   
**and**  $A6: \forall SQ\ TR. SQ \in S\ Q \wedge TR \in T\ R$   
 $\longrightarrow (\llbracket SQ \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A7: \forall TQ\ SR. TQ \in T\ Q \wedge SR \in S\ R \longrightarrow False$   
**and**  $A8: \forall TQ\ TR. TQ \in T\ Q \wedge TR \in T\ R \longrightarrow (TQ, TR) \in TRel^+$   
**show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow SP = SR$   
**proof** (*cases Q*)  
**case** (*SourceTerm SQ*)  
**assume**  $SQ \in S\ Q$   
**with**  $A1\ A5$  **show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow SP = SR$   
 by *blast*  
**next**  
**case** (*TargetTerm TQ*)  
**assume**  $TQ \in T\ Q$   
**with**  $A7$  **show** *?thesis*  
 by *blast*  
**qed**  
**show**  $\forall SP\ TR. SP \in S\ P \wedge TR \in T\ R$   
 $\longrightarrow (\llbracket SP \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** (*cases Q*)  
**case** (*SourceTerm SQ*)  
**assume**  $SQ \in S\ Q$   
**with**  $A1\ A6$  **show** *?thesis*  
 by *blast*  
**next**  
**case** (*TargetTerm TQ*)  
**assume**  $A9: TQ \in T\ Q$   
**show**  $\forall SP\ TR. SP \in S\ P \wedge TR \in T\ R$   
 $\longrightarrow (\llbracket SP \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** *clarify*  
**fix**  $SP\ TR$   
**assume**  $SP \in S\ P$   
**with**  $A2\ A9$  **have**  $(\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
 by *simp*  
**moreover** **assume**  $TR \in T\ R$   
**with**  $A8\ A9$  **have**  $(TQ, TR) \in TRel^+$   
 by *simp*  
**hence**  $(TQ, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$



```

proof induct
  fix  $T2$ 
  assume  $(TQ, T2) \in TRel$ 
  thus  $(TQ, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
    by blast
  next
  case (step  $T2$   $T3$ )
  assume  $(TQ, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  moreover assume  $(T2, T3) \in TRel$ 
  hence  $(T2, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
    by blast
  ultimately show  $(TQ, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
    by simp
  qed
  ultimately show  $(\llbracket SP \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
    by simp
  qed
qed
show  $\forall TP SR. TP \in T P \wedge SR \in S R \longrightarrow False$ 
proof (cases  $Q$ )
  case (SourceTerm  $SQ$ )
  assume  $SQ \in S Q$ 
  with  $A3$  show ?thesis
    by blast
  next
  case (TargetTerm  $TQ$ )
  assume  $TQ \in T Q$ 
  with  $A7$  show ?thesis
    by blast
  qed
show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
proof (cases  $Q$ )
  case (SourceTerm  $SQ$ )
  assume  $SQ \in S Q$ 
  with  $A3$  show ?thesis
    by blast
  next
  case (TargetTerm  $TQ$ )
  assume  $TQ \in T Q$ 
  with  $A4$   $A8$  show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
    by auto
  qed
qed
qed

```

**lemma** (*in encoding*) *indRelLTPO-to-TRel*:

```

fixes  $TRel :: ('procT \times 'procT) set$ 
and  $P Q :: ('procS, 'procT) Proc$ 
assumes rel:  $P \lesssim \llbracket \cdot \rrbracket LT < TRel > Q$ 
shows  $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow SP = SQ$ 
and  $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow False$ 
and  $\forall TP SQ. TP \in T P \wedge SQ \in S Q$ 
   $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
and  $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$ 
proof –
  have reflTRel:  $\forall S. (\llbracket S \rrbracket, \llbracket S \rrbracket) \in TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\}$ 
    by auto
  from rel show  $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow SP = SQ$ 
    and  $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow False$ 
    and  $\forall TP SQ. TP \in T P \wedge SQ \in S Q$ 
       $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 

```

**and**  $\forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$   
**proof** *induct*  
**case** (*encL S*)  
**show**  $\forall SP\ SQ. SP \in S\ TargetTerm\ (\llbracket S \rrbracket) \wedge SQ \in S\ SourceTerm\ S \longrightarrow SP = SQ$   
**and**  $\forall SP\ TQ. SP \in S\ TargetTerm\ (\llbracket S \rrbracket) \wedge TQ \in T\ SourceTerm\ S \longrightarrow False$   
**and**  $\forall TP\ TQ. TP \in T\ TargetTerm\ (\llbracket S \rrbracket) \wedge TQ \in T\ SourceTerm\ S \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp-all*  
**from** *reflTRel* **show**  $\forall TP\ SQ. TP \in T\ TargetTerm\ (\llbracket S \rrbracket) \wedge SQ \in S\ SourceTerm\ S$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *blast*  
**next**  
**case** (*source S*)  
**show**  $\forall SP\ SQ. SP \in S\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S \longrightarrow SP = SQ$   
**by** *simp*  
**show**  $\forall SP\ TQ. SP \in S\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S \longrightarrow False$   
**and**  $\forall TP\ SQ. TP \in T\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP\ TQ. TP \in T\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp-all*  
**next**  
**case** (*target T1 T2*)  
**show**  $\forall SP\ SQ. SP \in S\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2 \longrightarrow SP = SQ$   
**and**  $\forall SP\ TQ. SP \in S\ TargetTerm\ T1 \wedge TQ \in T\ TargetTerm\ T2 \longrightarrow False$   
**and**  $\forall TP\ SQ. TP \in T\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *simp-all*  
**assume**  $(T1, T2) \in TRel$   
**thus**  $\forall TP\ TQ. TP \in T\ TargetTerm\ T1 \wedge TQ \in T\ TargetTerm\ T2 \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp*  
**next**  
**case** (*trans P Q R*)  
**assume**  $A1: \forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow SP = SQ$   
**and**  $A2: \forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q \longrightarrow False$   
**and**  $A3: \forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A4: \forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$   
**and**  $A5: \forall SQ\ SR. SQ \in S\ Q \wedge SR \in S\ R \longrightarrow SQ = SR$   
**and**  $A6: \forall SQ\ TR. SQ \in S\ Q \wedge TR \in T\ R \longrightarrow False$   
**and**  $A7: \forall TQ\ SR. TQ \in T\ Q \wedge SR \in S\ R$   
 $\longrightarrow (TQ, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A8: \forall TQ\ TR. TQ \in T\ Q \wedge TR \in T\ R \longrightarrow (TQ, TR) \in TRel^+$   
**show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow SP = SR$   
**proof** (*cases Q*)  
**case** (*SourceTerm SQ*)  
**assume**  $SQ \in S\ Q$   
**with**  $A1\ A5$  **show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow SP = SR$   
**by** *blast*  
**next**  
**case** (*TargetTerm TQ*)  
**assume**  $TQ \in T\ Q$   
**with**  $A2$  **show** *?thesis*  
**by** *blast*  
**qed**  
**show**  $\forall SP\ TR. SP \in S\ P \wedge TR \in T\ R \longrightarrow False$   
**proof** (*cases Q*)  
**case** (*SourceTerm SQ*)  
**assume**  $SQ \in S\ Q$   
**with**  $A6$  **show** *?thesis*  
**by** *blast*  
**next**  
**case** (*TargetTerm TQ*)

```

assume  $TQ \in T Q$ 
with  $A2$  show ?thesis
  by blast
qed
show  $\forall TP SR. TP \in T P \wedge SR \in S R$ 
   $\longrightarrow (TP, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
proof (cases Q)
  case (SourceTerm SQ)
  assume  $SQ \in S Q$ 
  with  $A3 A5$  show  $\forall TP SR. TP \in T P \wedge SR \in S R$ 
     $\longrightarrow (TP, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
    by blast
next
  case (TargetTerm TQ)
  assume  $A9: TQ \in T Q$ 
  show  $\forall TP SR. TP \in T P \wedge SR \in S R$ 
     $\longrightarrow (TP, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  proof clarify
  fix  $TP SR$ 
  assume  $TP \in T P$ 
  with  $A4 A9$  have  $(TP, TQ) \in TRel^+$ 
  by simp
  hence  $(TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  proof induct
  fix  $T2$ 
  assume  $(TP, T2) \in TRel$ 
  thus  $(TP, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  by blast
next
  case (step T2 T3)
  assume  $(TP, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  moreover assume  $(T2, T3) \in TRel$ 
  hence  $(T2, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  by blast
  ultimately show  $(TP, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
  by simp
qed
moreover assume  $SR \in S R$ 
with  $A7 A9$  have  $(TQ, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
by simp
ultimately show  $(TP, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$ 
by simp
qed
qed
show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
proof (cases Q)
  case (SourceTerm SQ)
  assume  $SQ \in S Q$ 
  with  $A6$  show ?thesis
  by blast
next
  case (TargetTerm TQ)
  assume  $TQ \in T Q$ 
  with  $A4 A8$  show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
  by auto
qed
qed
qed

```

**lemma** (*in encoding*) *indRelTEQ-to-TRel*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$

**and**  $P Q :: ('procS, 'procT) Proc$   
**assumes**  $rel: P \sim \llbracket \cdot \rrbracket T < TRel > Q$   
**shows**  $\forall SP SQ. SP \in S P \wedge SQ \in S Q$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall SP TQ. SP \in S P \wedge TQ \in T Q$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP SQ. TP \in T P \wedge SQ \in S Q$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP TQ. TP \in T P \wedge TQ \in T Q$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** –  
**have**  $reflTRel: \forall S. (\llbracket S \rrbracket, \llbracket S \rrbracket) \in TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\}$   
**by** *auto*  
**from**  $rel$  **show**  $\forall SP SQ. SP \in S P \wedge SQ \in S Q$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall SP TQ. SP \in S P \wedge TQ \in T Q$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP SQ. TP \in T P \wedge SQ \in S Q$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP TQ. TP \in T P \wedge TQ \in T Q$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** *induct*  
**case** (*encR S*)  
**show**  $\forall SP SQ. SP \in S SourceTerm S \wedge SQ \in S TargetTerm (\llbracket S \rrbracket)$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP SQ. TP \in T SourceTerm S \wedge SQ \in S TargetTerm (\llbracket S \rrbracket)$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP TQ. TP \in T SourceTerm S \wedge TQ \in T TargetTerm (\llbracket S \rrbracket)$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *simp+*  
**from**  $reflTRel$  **show**  $\forall SP TQ. SP \in S SourceTerm S \wedge TQ \in T TargetTerm (\llbracket S \rrbracket)$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *blast*  
**next**  
**case** (*encL S*)  
**show**  $\forall SP SQ. SP \in S TargetTerm (\llbracket S \rrbracket) \wedge SQ \in S SourceTerm S$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall SP TQ. SP \in S TargetTerm (\llbracket S \rrbracket) \wedge TQ \in T SourceTerm S$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP TQ. TP \in T TargetTerm (\llbracket S \rrbracket) \wedge TQ \in T SourceTerm S$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *simp+*  
**from**  $reflTRel$  **show**  $\forall TP SQ. TP \in T TargetTerm (\llbracket S \rrbracket) \wedge SQ \in S SourceTerm S$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**show**  $\forall SP SQ. SP \in S TargetTerm T1 \wedge SQ \in S TargetTerm T2$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall SP TQ. SP \in S TargetTerm T1 \wedge TQ \in T TargetTerm T2$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $\forall TP SQ. TP \in T TargetTerm T1 \wedge SQ \in S TargetTerm T2$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *simp+*  
**assume**  $(T1, T2) \in TRel$   
**thus**  $\forall TP TQ. TP \in T TargetTerm T1 \wedge TQ \in T TargetTerm T2$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**by** *blast*  
**next**  
**case** (*trans P Q R*)  
**assume**  $A1: \forall SP SQ. SP \in S P \wedge SQ \in S Q$

$\longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A2: \forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q$   
 $\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A3: \forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q$   
 $\longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A4: \forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q$   
 $\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A5: \forall SQ\ SR. SQ \in S\ Q \wedge SR \in S\ R$   
 $\longrightarrow (\llbracket SQ \rrbracket, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A6: \forall SQ\ TR. SQ \in S\ Q \wedge TR \in T\ R$   
 $\longrightarrow (\llbracket SQ \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A7: \forall TQ\ SR. TQ \in T\ Q \wedge SR \in S\ R$   
 $\longrightarrow (TQ, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**and**  $A8: \forall TQ\ TR. TQ \in T\ Q \wedge TR \in T\ R$   
 $\longrightarrow (TQ, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R$   
 $\longrightarrow (\llbracket SP \rrbracket, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $SQ \in S\ Q$   
**with**  $A1\ A5$  **show** *?thesis*  
**by** *auto*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $TQ \in T\ Q$   
**with**  $A2\ A7$  **show** *?thesis*  
**by** *auto*  
**qed**  
**show**  $\forall SP\ TR. SP \in S\ P \wedge TR \in T\ R \longrightarrow (\llbracket SP \rrbracket, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $SQ \in S\ Q$   
**with**  $A1\ A6$  **show** *?thesis*  
**by** *auto*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $TQ \in T\ Q$   
**with**  $A2\ A8$  **show** *?thesis*  
**by** *auto*  
**qed**  
**show**  $\forall TP\ SR. TP \in T\ P \wedge SR \in S\ R \longrightarrow (TP, \llbracket SR \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $SQ \in S\ Q$   
**with**  $A3\ A5$  **show** *?thesis*  
**by** *auto*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $TQ \in T\ Q$   
**with**  $A4\ A7$  **show** *?thesis*  
**by** *auto*  
**qed**  
**show**  $\forall TP\ TR. TP \in T\ P \wedge TR \in T\ R \longrightarrow (TP, TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $SQ \in S\ Q$   
**with**  $A3\ A6$  **show** *?thesis*  
**by** *auto*

```

next
  case (TargetTerm TQ)
  assume TQ ∈ T Q
  with A4 A8 show ?thesis
  by auto
qed
qed
qed

```

```

lemma (in encoding) trans-closure-of-TRel-refl-cond:
  fixes TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  assumes (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})+
  shows (TP, TQ) ∈ TRel*
  using assms
proof induct
  fix TQ
  assume (TP, TQ) ∈ TRel ∪ {(T1, T2). ∃ S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧}
  thus (TP, TQ) ∈ TRel*
  by auto
next
  case (step TQ TR)
  assume (TP, TQ) ∈ TRel*
  moreover assume (TQ, TR) ∈ TRel ∪ {(T1, T2). ∃ S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧}
  hence (TQ, TR) ∈ TRel*
  by blast
  ultimately show (TP, TR) ∈ TRel*
  by simp
qed

```

Note that if  $\text{indRelRTPO}$  relates a source term  $S$  to a target term  $T$ , then the translation of  $S$  is equal to  $T$  or  $\text{indRelRTPO}$  also relates the translation of  $S$  to  $T$ .

```

lemma (in encoding) indRelRTPO-relates-source-target:
  fixes TRel :: ('procT × 'procT) set
  and S :: 'procS
  and T :: 'procT
  assumes pair: SourceTerm S ≲[·]RT<TRel> TargetTerm T
  shows (TargetTerm (⟦S⟧), TargetTerm T) ∈ (indRelRTPO TRel)=
proof -
  from pair have (⟦S⟧, T) ∈ TRel*
  using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond
  by simp
  hence ⟦S⟧ = T ∨ (⟦S⟧, T) ∈ TRel+
  using rtrancl-eq-or-trancl[of ⟦S⟧ T TRel]
  by blast
  moreover have ⟦S⟧ = T ⇒ (TargetTerm (⟦S⟧), TargetTerm T) ∈ (indRelRTPO TRel)=
  by simp
  moreover
  have (⟦S⟧, T) ∈ TRel+ ⇒ (TargetTerm (⟦S⟧), TargetTerm T) ∈ (indRelRTPO TRel)=
  using transitive-closure-of-TRel-to-indRelRTPO[where TRel=TRel]
  by simp
  ultimately show (TargetTerm (⟦S⟧), TargetTerm T) ∈ (indRelRTPO TRel)=
  by blast
qed

```

If  $\text{indRelRTPO}$ ,  $\text{indRelLTPO}$ , or  $\text{indRelTPO}$  preserves barbs then so does the corresponding target term relation.

```

lemma (in encoding-wrt-barbs) rel-with-target-impl-TRel-preserves-barbs:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set

```

**assumes** *preservation*: *rel-preserves-barbs* *Rel* (*STCalWB SWB TWB*)  
**and** *targetInRel*:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**shows** *rel-preserves-barbs* *TRel TWB*  
**proof** *clarify*  
**fix** *TP TQ a*  
**assume**  $(TP, TQ) \in TRel$   
**with** *targetInRel* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *blast*  
**moreover** **assume**  $TP \downarrow \langle TWB \rangle a$   
**hence** *TargetTerm*  $TP \downarrow .a$   
**by** *simp*  
**ultimately** **have** *TargetTerm*  $TQ \downarrow .a$   
**using** *preservation preservation-of-barbs-in-barbed-encoding* [**where** *Rel=Rel*]  
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle a$   
**by** *simp*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-preserves-barbs*:  
**fixes** *TRel* ::  $('procT \times 'procT)$  *set*  
**assumes** *preservation*: *rel-preserves-barbs* (*indRelRTPO TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-preserves-barbs* *TRel TWB*  
**using** *preservation*  
*rel-with-target-impl-TRel-preserves-barbs* [**where** *Rel=indRelRTPO TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelRTPO.target*)

**lemma** (**in** *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-preserves-barbs*:  
**fixes** *TRel* ::  $('procT \times 'procT)$  *set*  
**assumes** *preservation*: *rel-preserves-barbs* (*indRelLTPO TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-preserves-barbs* *TRel TWB*  
**using** *preservation*  
*rel-with-target-impl-TRel-preserves-barbs* [**where** *Rel=indRelLTPO TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelLTPO.target*)

**lemma** (**in** *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-preserves-barbs*:  
**fixes** *TRel* ::  $('procT \times 'procT)$  *set*  
**assumes** *preservation*: *rel-preserves-barbs* (*indRelTEQ TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-preserves-barbs* *TRel TWB*  
**using** *preservation*  
*rel-with-target-impl-TRel-preserves-barbs* [**where** *Rel=indRelTEQ TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelTEQ.target*)

**lemma** (**in** *encoding-wrt-barbs*) *rel-with-target-impl-TRel-weakly-preserves-barbs*:  
**fixes** *TRel* ::  $('procT \times 'procT)$  *set*  
**and** *Rel* ::  $(('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$  *set*  
**assumes** *preservation*: *rel-weakly-preserves-barbs* *Rel* (*STCalWB SWB TWB*)  
**and** *targetInRel*:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**shows** *rel-weakly-preserves-barbs* *TRel TWB*  
**proof** *clarify*  
**fix** *TP TQ a TP'*  
**assume**  $(TP, TQ) \in TRel$   
**with** *targetInRel* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *blast*  
**moreover** **assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \downarrow \langle TWB \rangle a$   
**hence** *TargetTerm*  $TP \downarrow .a$   
**by** *blast*  
**ultimately** **have** *TargetTerm*  $TQ \downarrow .a$   
**using** *preservation weak-preservation-of-barbs-in-barbed-encoding* [**where** *Rel=Rel*]  
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle a$   
**by** *simp*

qed

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-preserves-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *preservation: rel-weakly-preserves-barbs* (*indRelRTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
shows *rel-weakly-preserves-barbs*  $TRel$   $TWB$   
using *preservation rel-with-target-impl-TRel-weakly-preserves-barbs* [where  
 $Rel = indRelRTPO\ TRel$  and  $TRel = TRel$ ]  
by (*simp add: indRelRTPO.target*)

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-weakly-preserves-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *preservation: rel-weakly-preserves-barbs* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
shows *rel-weakly-preserves-barbs*  $TRel$   $TWB$   
using *preservation rel-with-target-impl-TRel-weakly-preserves-barbs* [where  
 $Rel = indRelLTPO\ TRel$  and  $TRel = TRel$ ]  
by (*simp add: indRelLTPO.target*)

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-preserves-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *preservation: rel-weakly-preserves-barbs* (*indRelTEQ*  $TRel$ ) (*STCalWB SWB TWB*)  
shows *rel-weakly-preserves-barbs*  $TRel$   $TWB$   
using *preservation rel-with-target-impl-TRel-weakly-preserves-barbs* [where  
 $Rel = indRelTEQ\ TRel$  and  $TRel = TRel$ ]  
by (*simp add: indRelTEQ.target*)

If *indRelRTPO*, *indRelLTPO*, or *indRelTPO* reflects barbs then so does the corresponding target term relation.

**lemma** (in *encoding-wrt-barbs*) *rel-with-target-impl-TRel-reflects-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
and  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
assumes *reflection: rel-reflects-barbs*  $Rel$  (*STCalWB SWB TWB*)  
and *targetInRel:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$*   
shows *rel-reflects-barbs*  $TRel$   $TWB$

**proof** *clarify*  
fix  $TP\ TQ\ a$   
assume  $(TP, TQ) \in TRel$   
with *targetInRel* have  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
by *blast*  
moreover assume  $TQ \downarrow < TWB > a$   
hence  $TargetTerm\ TQ \downarrow . a$   
by *simp*  
ultimately have  $TargetTerm\ TP \downarrow . a$   
using *reflection reflection-of-barbs-in-barbed-encoding* [where  $Rel = Rel$ ]  
by *blast*  
thus  $TP \downarrow < TWB > a$   
by *simp*

qed

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-reflects-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *reflection: rel-reflects-barbs* (*indRelRTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
shows *rel-reflects-barbs*  $TRel$   $TWB$   
using *reflection*  
*rel-with-target-impl-TRel-reflects-barbs* [where  $Rel = indRelRTPO\ TRel$  and  $TRel = TRel$ ]  
by (*simp add: indRelRTPO.target*)

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-reflects-barbs*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *reflection: rel-reflects-barbs* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)



**shows** *rel-reflects-barbs* *TRel* *TWB*  
**using** *reflection*  
*rel-with-target-impl-TRel-reflects-barbs*[**where** *Rel=indRelLTPO* *TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelLTPO.target*)

**lemma** (**in** *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-reflects-barbs*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**assumes** *reflection: rel-reflects-barbs* (*indRelTEQ* *TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-reflects-barbs* *TRel* *TWB*  
**using** *reflection*  
*rel-with-target-impl-TRel-reflects-barbs*[**where** *Rel=indRelTEQ* *TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelTEQ.target*)

**lemma** (**in** *encoding-wrt-barbs*) *rel-with-target-impl-TRel-weakly-reflects-barbs*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**and** *Rel* :: (('proc*S*, '*procT*) *Proc* × ('proc*S*, '*procT*) *Proc*) *set*  
**assumes** *reflection: rel-weakly-reflects-barbs* *Rel* (*STCalWB SWB TWB*)  
**and** *targetInRel: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel*  
**shows** *rel-weakly-reflects-barbs* *TRel* *TWB*

**proof** *clarify*  
**fix** *TP TQ a TQ'*  
**assume** (*TP, TQ*) ∈ *TRel*  
**with** *targetInRel* **have** (*TargetTerm TP, TargetTerm TQ*) ∈ *Rel*  
**by** *blast*  
**moreover** **assume** *TQ*  $\mapsto$  (*Calculus* *TWB*)\* *TQ'* **and** *TQ' ↓ < TWB > a*  
**hence** *TargetTerm TQ ↓ . a*  
**by** *blast*  
**ultimately** **have** *TargetTerm TP ↓ . a*  
**using** *reflection weak-reflection-of-barbs-in-barbed-encoding*[**where** *Rel=Rel*]  
**by** *blast*  
**thus** *TP ↓ < TWB > a*  
**by** *simp*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-reflects-barbs*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**assumes** *reflection: rel-weakly-reflects-barbs* (*indRelRTPO* *TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-reflects-barbs* *TRel* *TWB*  
**using** *reflection rel-with-target-impl-TRel-weakly-reflects-barbs*[**where**  
*Rel=indRelRTPO* *TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelRTPO.target*)

**lemma** (**in** *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-weakly-reflects-barbs*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**assumes** *reflection: rel-weakly-reflects-barbs* (*indRelLTPO* *TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-reflects-barbs* *TRel* *TWB*  
**using** *reflection rel-with-target-impl-TRel-weakly-reflects-barbs*[**where**  
*Rel=indRelLTPO* *TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelLTPO.target*)

**lemma** (**in** *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-reflects-barbs*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**assumes** *reflection: rel-weakly-reflects-barbs* (*indRelTEQ* *TRel*) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-reflects-barbs* *TRel* *TWB*  
**using** *reflection rel-with-target-impl-TRel-weakly-reflects-barbs*[**where**  
*Rel=indRelTEQ* *TRel* **and** *TRel=TRel*]  
**by** (*simp add: indRelTEQ.target*)

If *indRelRTPO*, *indRelLTPO*, or *indRelTPO* respects *barbs* then so does the corresponding target term relation.

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-respects-barbs* (*indRelRTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelRTPO-impl-TRel-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelRTPO-impl-TRel-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-respects-barbs* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelLTPO-impl-TRel-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelLTPO-impl-TRel-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-respects-barbs* (*indRelTEQ*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelTEQ-impl-TRel-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelTEQ-impl-TRel-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-weakly-respects-barbs* (*indRelRTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelRTPO-impl-TRel-weakly-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelRTPO-impl-TRel-weakly-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-weakly-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-weakly-respects-barbs* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelLTPO-impl-TRel-weakly-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelLTPO-impl-TRel-weakly-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-respects-barbs*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-weakly-respects-barbs* (*indRelTEQ*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelTEQ-impl-TRel-weakly-preserves-barbs*[**where**  $TRel=TRel$ ]  
*indRelTEQ-impl-TRel-weakly-reflects-barbs*[**where**  $TRel=TRel$ ]  
**by** *blast*

If *indRelRTPO*, *indRelLTPO*, or *indRelTEQ* is a simulation then so is the corresponding target term relation.

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *sim: weak-reduction-simulation*  $Rel$  (*STCal Source Target*)  
**and** *target:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*   
**and** *trel:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$*   
**shows** *weak-reduction-simulation* ( $TRel^+$ )  $Target$

**proof** *clarify*  
**fix**  $TP TQ TP'$   
**assume**  $(TP, TQ) \in TRel^+$  **and**  $TP \longmapsto Target* TP'$

**thus**  $\exists TQ'. TQ \mapsto \text{Target}^* TQ' \wedge (TP', TQ') \in TRel^+$   
**proof** (*induct arbitrary: TP'*)  
**fix**  $TQ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with target have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in Rel$   
**by simp**  
**moreover assume**  $TP \mapsto \text{Target}^* TP'$   
**hence**  $\text{TargetTerm } TP \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TP')$   
**by** (*simp add: STCal-steps*)  
**ultimately obtain**  $Q'$  **where**  $A2: \text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* Q'$   
**and**  $A3: (\text{TargetTerm } TP', Q') \in Rel$   
**using sim**  
**by blast**  
**from**  $A2$  **obtain**  $TQ'$  **where**  $A4: TQ \mapsto \text{Target}^* TQ'$  **and**  $A5: TQ' \in T Q'$   
**by** (*auto simp add: STCal-steps*)  
**from**  $A3 A5$  **trel have**  $(TP', TQ') \in TRel^+$   
**by simp**  
**with**  $A4$  **show**  $\exists TQ'. TQ \mapsto \text{Target}^* TQ' \wedge (TP', TQ') \in TRel^+$   
**by blast**  
**next**  
**case** (*step TQ TR*)  
**assume**  $TP \mapsto \text{Target}^* TP'$   
**and**  $\bigwedge TP'. TP \mapsto \text{Target}^* TP' \implies \exists TQ'. TQ \mapsto \text{Target}^* TQ' \wedge (TP', TQ') \in TRel^+$   
**from this obtain**  $TQ'$  **where**  $B1: TQ \mapsto \text{Target}^* TQ'$  **and**  $B2: (TP', TQ') \in TRel^+$   
**by blast**  
**assume**  $(TQ, TR) \in TRel$   
**with target have**  $(\text{TargetTerm } TQ, \text{TargetTerm } TR) \in Rel$   
**by simp**  
**moreover from**  $B1$  **have**  $\text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TQ')$   
**by** (*simp add: STCal-steps*)  
**ultimately obtain**  $R'$  **where**  $B3: \text{TargetTerm } TR \mapsto (\text{STCal Source Target})^* R'$   
**and**  $B4: (\text{TargetTerm } TQ', R') \in Rel$   
**using sim**  
**by blast**  
**from**  $B3$  **obtain**  $TR'$  **where**  $B5: TR' \in T R'$  **and**  $B6: TR \mapsto \text{Target}^* TR'$   
**by** (*auto simp add: STCal-steps*)  
**from**  $B4 B5$  **trel have**  $(TQ', TR') \in TRel^+$   
**by simp**  
**with**  $B2$  **have**  $(TP', TR') \in TRel^+$   
**by simp**  
**with**  $B6$  **show**  $\exists TR'. TR \mapsto \text{Target}^* TR' \wedge (TP', TR') \in TRel^+$   
**by blast**  
**qed**  
**qed**

**lemma** (*in encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *sim: weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)*  
**shows** *weak-reduction-simulation (TRel<sup>+</sup>) Target*  
**using** *sim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel(4)[where TRel=TRel]*  
*rel-with-target-impl-transC-TRel-is-weak-reduction-simulation[where*  
*Rel=indRelRTPO TRel and TRel=TRel]*  
**by blast**

**lemma** (*in encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *sim: weak-reduction-simulation (indRelLTPO TRel) (STCal Source Target)*  
**shows** *weak-reduction-simulation (TRel<sup>+</sup>) Target*  
**using** *sim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(4)[where TRel=TRel]*  
*rel-with-target-impl-transC-TRel-is-weak-reduction-simulation[where*  
*Rel=indRelLTPO TRel and TRel=TRel]*

by *blast*

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev*:

**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$

**and**  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$

**assumes** *sim*: *weak-reduction-simulation*  $(Rel^{-1})$   $(STCal \text{ Source Target})$

**and** *target*:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$

**and** *trel*:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$

**shows** *weak-reduction-simulation*  $((TRel^+)^{-1})$  *Target*

**proof** *clarify*

**fix**  $TP \ TQ \ TP'$

**assume**  $(TQ, TP) \in TRel^+$

**moreover assume**  $TP \longmapsto Target^* TP'$

**ultimately show**  $\exists TQ'. TQ \longmapsto Target^* TQ' \wedge (TP', TQ') \in (TRel^+)^{-1}$

**proof** (*induct arbitrary*:  $TP'$ )

**fix**  $TP \ TP'$

**assume**  $(TQ, TP) \in TRel$

**with target have**  $(TargetTerm TP, TargetTerm TQ) \in Rel^{-1}$

**by** *simp*

**moreover assume**  $TP \longmapsto Target^* TP'$

**hence**  $TargetTerm TP \longmapsto (STCal \text{ Source Target})^* (TargetTerm TP')$

**by** (*simp add*: *STCal-steps*)

**ultimately obtain**  $Q'$  **where**  $A2: TargetTerm TQ \longmapsto (STCal \text{ Source Target})^* Q'$

**and**  $A3: (TargetTerm TP', Q') \in Rel^{-1}$

**using** *sim*

**by** *blast*

**from**  $A2$  **obtain**  $TQ'$  **where**  $A4: TQ \longmapsto Target^* TQ'$  **and**  $A5: TQ' \in T \ Q'$

**by** (*auto simp add*: *STCal-steps(2)*)

**from**  $A3 \ A5$  **trel have**  $(TP', TQ') \in (TRel^+)^{-1}$

**by** *simp*

**with**  $A4$  **show**  $\exists TQ'. TQ \longmapsto Target^* TQ' \wedge (TP', TQ') \in (TRel^+)^{-1}$

**by** *blast*

**next**

**case** (*step TR TP TP'*)

**assume**  $TP \longmapsto Target^* TP'$

**hence**  $TargetTerm TP \longmapsto (STCal \text{ Source Target})^* (TargetTerm TP')$

**by** (*simp add*: *STCal-steps*)

**moreover assume**  $(TR, TP) \in TRel$

**with target have**  $(TargetTerm TP, TargetTerm TR) \in Rel^{-1}$

**by** *simp*

**ultimately obtain**  $R'$  **where**  $B1: TargetTerm TR \longmapsto (STCal \text{ Source Target})^* R'$

**and**  $B2: (TargetTerm TP', R') \in Rel^{-1}$

**using** *sim*

**by** *blast*

**from**  $B1$  **obtain**  $TR'$  **where**  $B3: TR' \in T \ R'$  **and**  $B4: TR \longmapsto Target^* TR'$

**by** (*auto simp add*: *STCal-steps*)

**assume**  $\bigwedge TR'. TR \longmapsto Target^* TR' \implies \exists TQ'. TQ \longmapsto Target^* TQ' \wedge (TR', TQ') \in (TRel^+)^{-1}$

**with**  $B4$  **obtain**  $TQ'$  **where**  $B5: TQ \longmapsto Target^* TQ'$  **and**  $B6: (TR', TQ') \in (TRel^+)^{-1}$

**by** *blast*

**from**  $B6$  **have**  $(TQ', TR') \in TRel^+$

**by** *simp*

**moreover from**  $B2 \ B3$  **trel have**  $(TR', TP') \in TRel^+$

**by** *simp*

**ultimately have**  $(TP', TQ') \in (TRel^+)^{-1}$

**by** *simp*

**with**  $B5$  **show**  $\exists TQ'. TQ \longmapsto Target^* TQ' \wedge (TP', TQ') \in (TRel^+)^{-1}$

**by** *blast*

**qed**

**qed**

**lemma** (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev*:

**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes**  $sim: \text{weak-reduction-simulation } ((indRelRTPO \ TRel)^{-1}) \ (STCal \ Source \ Target)$   
**shows**  $\text{weak-reduction-simulation } ((TRel^+)^{-1}) \ Target$   
**using**  $sim \ indRelRTPO.target[\mathbf{where} \ TRel=TRel] \ indRelRTPO\text{-to-}TRel(4)[\mathbf{where} \ TRel=TRel]$   
 $\text{rel-with-target-impl-transC-}TRel\text{-is-weak-reduction-simulation-rev}[\mathbf{where}$   
 $\text{Rel}=\mathit{indRelRTPO} \ TRel \ \mathbf{and} \ TRel=TRel]$   
**by**  $blast$

**lemma** (*in encoding*)  $\text{indRelLTPO-impl-}TRel\text{-is-weak-reduction-simulation-rev}$ :  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes**  $sim: \text{weak-reduction-simulation } ((indRelLTPO \ TRel)^{-1}) \ (STCal \ Source \ Target)$   
**shows**  $\text{weak-reduction-simulation } ((TRel^+)^{-1}) \ Target$   
**using**  $sim \ indRelLTPO.target[\mathbf{where} \ TRel=TRel] \ indRelLTPO\text{-to-}TRel(4)[\mathbf{where} \ TRel=TRel]$   
 $\text{rel-with-target-impl-transC-}TRel\text{-is-weak-reduction-simulation-rev}[\mathbf{where}$   
 $\text{Rel}=\mathit{indRelLTPO} \ TRel \ \mathbf{and} \ TRel=TRel]$   
**by**  $blast$

**lemma** (*in encoding*)  $\text{rel-with-target-impl-reflC-transC-}TRel\text{-is-weak-reduction-simulation}$ :  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \ Proc \times ('procS, 'procT) \ Proc) \text{ set}$   
**assumes**  $sim: \text{weak-reduction-simulation } Rel \ (STCal \ Source \ Target)$   
**and**  $target: \forall T1 \ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in Rel$   
**and**  $trel: \forall T1 \ T2. (TargetTerm \ T1, TargetTerm \ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$   
**shows**  $\text{weak-reduction-simulation } (TRel^*) \ Target$

**proof**  $clarify$   
**fix**  $TP \ TQ \ TP'$   
**assume**  $(TP, TQ) \in TRel^* \ \mathbf{and} \ TP \longmapsto Target^* \ TP'$   
**thus**  $\exists TQ'. TQ \longmapsto Target^* \ TQ' \wedge (TP', TQ') \in TRel^*$   
**proof** (*induct arbitrary: TP'*)  
**fix**  $TP'$   
**assume**  $TP \longmapsto Target^* \ TP'$   
**moreover** **have**  $(TP', TP') \in TRel^*$   
**by**  $simp$   
**ultimately** **show**  $\exists TQ'. TP \longmapsto Target^* \ TQ' \wedge (TP', TQ') \in TRel^*$   
**by**  $blast$

**next**

**case** ( $step \ TQ \ TR$ )  
**assume**  $TP \longmapsto Target^* \ TP'$   
**and**  $\bigwedge TP'. TP \longmapsto Target^* \ TP' \implies \exists TQ'. TQ \longmapsto Target^* \ TQ' \wedge (TP', TQ') \in TRel^*$   
**from** *this* **obtain**  $TQ'$  **where**  $B1: TQ \longmapsto Target^* \ TQ'$  **and**  $B2: (TP', TQ') \in TRel^*$   
**by**  $blast$   
**assume**  $(TQ, TR) \in TRel$   
**with**  $target$  **have**  $(TargetTerm \ TQ, TargetTerm \ TR) \in Rel$   
**by**  $simp$   
**moreover** **from**  $B1$  **have**  $TargetTerm \ TQ \longmapsto (STCal \ Source \ Target)^* \ (TargetTerm \ TQ')$   
**by** ( $simp \ add: STCal\text{-steps}$ )  
**ultimately** **obtain**  $R'$  **where**  $B3: TargetTerm \ TR \longmapsto (STCal \ Source \ Target)^* \ R'$   
**and**  $B4: (TargetTerm \ TQ', R') \in Rel$   
**using**  $sim$   
**by**  $blast$   
**from**  $B3$  **obtain**  $TR'$  **where**  $B5: TR' \in T \ R'$  **and**  $B6: TR \longmapsto Target^* \ TR'$   
**by** ( $auto \ simp \ add: STCal\text{-steps}$ )  
**from**  $B4 \ B5 \ trel$  **have**  $(TQ', TR') \in TRel^*$   
**by**  $simp$   
**with**  $B2$  **have**  $(TP', TR') \in TRel^*$   
**by**  $simp$   
**with**  $B6$  **show**  $\exists TR'. TR \longmapsto Target^* \ TR' \wedge (TP', TR') \in TRel^*$   
**by**  $blast$

**qed**

**qed**

**lemma** (in *encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *sim*: *weak-reduction-simulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
**shows** *weak-reduction-simulation* ( $TRel^*$ ) *Target*  
**using** *sim* *indRelTEQ.target*[**where**  $TRel = TRel$ ] *indRelTEQ-to-TRel(4)*[**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*  
*rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation*[**where**  
 $Rel = indRelTEQ\ TRel$  **and**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *sim*: *strong-reduction-simulation*  $Rel$  (*STCal* *Source* *Target*)  
**and** *target*:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and** *trel*:  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**shows** *strong-reduction-simulation* ( $TRel^+$ ) *Target*

**proof** *clarify*  
**fix**  $TP\ TQ\ TP'$   
**assume**  $(TP, TQ) \in TRel^+$  **and**  $TP \mapsto Target\ TP'$   
**thus**  $\exists TQ'. TQ \mapsto Target\ TQ' \wedge (TP', TQ') \in TRel^+$   
**proof** (*induct arbitrary: TP'*)  
**fix**  $TQ\ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with** *target* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TP \mapsto Target\ TP'$   
**hence**  $TargetTerm\ TP \mapsto (STCal\ Source\ Target)\ (TargetTerm\ TP')$   
**by** (*simp add: STCal-step*)  
**ultimately** **obtain**  $Q'$  **where**  $A2: TargetTerm\ TQ \mapsto (STCal\ Source\ Target)\ Q'$   
**and**  $A3: (TargetTerm\ TP', Q') \in Rel$   
**using** *sim*  
**by** *blast*  
**from**  $A2$  **obtain**  $TQ'$  **where**  $A4: TQ \mapsto Target\ TQ'$  **and**  $A5: TQ' \in T\ Q'$   
**by** (*auto simp add: STCal-step*)  
**from**  $A3\ A5$  **trel** **have**  $(TP', TQ') \in TRel^+$   
**by** *simp*  
**with**  $A4$  **show**  $\exists TQ'. TQ \mapsto Target\ TQ' \wedge (TP', TQ') \in TRel^+$   
**by** *blast*

**next**

**case** (*step*  $TQ\ TR$ )  
**assume**  $TP \mapsto Target\ TP'$   
**and**  $\bigwedge TP'. TP \mapsto Target\ TP' \implies \exists TQ'. TQ \mapsto Target\ TQ' \wedge (TP', TQ') \in TRel^+$   
**from** *this* **obtain**  $TQ'$  **where**  $B1: TQ \mapsto Target\ TQ'$  **and**  $B2: (TP', TQ') \in TRel^+$   
**by** *blast*  
**assume**  $(TQ, TR) \in TRel$   
**with** *target* **have**  $(TargetTerm\ TQ, TargetTerm\ TR) \in Rel$   
**by** *simp*  
**moreover** **from**  $B1$  **have**  $TargetTerm\ TQ \mapsto (STCal\ Source\ Target)\ (TargetTerm\ TQ')$   
**by** (*simp add: STCal-step*)  
**ultimately** **obtain**  $R'$  **where**  $B3: TargetTerm\ TR \mapsto (STCal\ Source\ Target)\ R'$   
**and**  $B4: (TargetTerm\ TQ', R') \in Rel$   
**using** *sim*  
**by** *blast*  
**from**  $B3$  **obtain**  $TR'$  **where**  $B5: TR' \in T\ R'$  **and**  $B6: TR \mapsto Target\ TR'$   
**by** (*auto simp add: STCal-step*)  
**from**  $B4\ B5$  **trel** **have**  $(TQ', TR') \in TRel^+$   
**by** *simp*  
**with**  $B2$  **have**  $(TP', TR') \in TRel^+$   
**by** *simp*  
**with**  $B6$  **show**  $\exists TR'. TR \mapsto Target\ TR' \wedge (TP', TR') \in TRel^+$

by blast  
qed  
qed

**lemma** (in encoding) *indRelRTPO-impl-TRel-is-strong-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *sim*: strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)  
**shows** strong-reduction-simulation (TRel<sup>+</sup>) Target  
**using** *sim* indRelRTPO.target[**where** TRel=TRel] indRelRTPO-to-TRel(4)[**where** TRel=TRel]  
rel-with-target-impl-transC-TRel-is-strong-reduction-simulation[**where**  
Rel=indRelRTPO TRel **and** TRel=TRel]  
**by** blast

**lemma** (in encoding) *indRelLTPO-impl-TRel-is-strong-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *sim*: strong-reduction-simulation (indRelLTPO TRel) (STCal Source Target)  
**shows** strong-reduction-simulation (TRel<sup>+</sup>) Target  
**using** *sim* indRelLTPO.target[**where** TRel=TRel] indRelLTPO-to-TRel(4)[**where** TRel=TRel]  
rel-with-target-impl-transC-TRel-is-strong-reduction-simulation[**where**  
Rel=indRelLTPO TRel **and** TRel=TRel]  
**by** blast

**lemma** (in encoding) *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *sim*: strong-reduction-simulation (Rel<sup>-1</sup>) (STCal Source Target)  
**and** *target*:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**and** *trel*:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**shows** strong-reduction-simulation ((TRel<sup>+</sup>)<sup>-1</sup>) Target

**proof** clarify  
**fix**  $TP TQ TP'$   
**assume**  $(TQ, TP) \in TRel^+$   
**moreover assume**  $TP \longmapsto Target TP'$   
**ultimately show**  $\exists TQ'. TQ \longmapsto Target TQ' \wedge (TP', TQ') \in (TRel^+)^{-1}$   
**proof** (induct arbitrary: TP')  
**fix**  $TP TP'$   
**assume**  $(TQ, TP) \in TRel$   
**with target have**  $(TargetTerm TP, TargetTerm TQ) \in Rel^{-1}$   
**by simp**  
**moreover assume**  $TP \longmapsto Target TP'$   
**hence**  $TargetTerm TP \longmapsto (STCal Source Target) (TargetTerm TP')$   
**by (simp add: STCal-step)**  
**ultimately obtain**  $Q'$  **where**  $A2: TargetTerm TQ \longmapsto (STCal Source Target) Q'$   
**and**  $A3: (TargetTerm TP', Q') \in Rel^{-1}$   
**using** *sim*  
**by blast**  
**from**  $A2$  **obtain**  $TQ'$  **where**  $A4: TQ \longmapsto Target TQ'$  **and**  $A5: TQ' \in T Q'$   
**by (auto simp add: STCal-step(2))**  
**from**  $A3 A5 trel$  **have**  $(TP', TQ') \in (TRel^+)^{-1}$   
**by simp**  
**with**  $A4$  **show**  $\exists TQ'. TQ \longmapsto Target TQ' \wedge (TP', TQ') \in (TRel^+)^{-1}$   
**by blast**

**next**

**case** (step TP TR TR')  
**assume**  $(TP, TR) \in TRel$   
**with target have**  $(TargetTerm TP, TargetTerm TR) \in Rel$   
**by simp**  
**moreover assume**  $TR \longmapsto Target TR'$   
**hence**  $TargetTerm TR \longmapsto (STCal Source Target) (TargetTerm TR')$   
**by (simp add: STCal-step)**  
**ultimately obtain**  $P'$  **where**  $B1: TargetTerm TP \longmapsto (STCal Source Target) P'$

and  $B2: (P', \text{TargetTerm } TR') \in \text{Rel}$

using *sim*

by *blast*

from  $B1$  obtain  $TP'$  where  $B3: TP' \in T P'$  and  $B4: TP \mapsto \text{Target } TP'$

by (*auto simp add: STCal-step*)

assume  $\bigwedge TP'. TP \mapsto \text{Target } TP' \implies \exists TQ'. TQ \mapsto \text{Target } TQ' \wedge (TP', TQ') \in (\text{TRel}^+)^{-1}$

with  $B4$  obtain  $TQ'$  where  $B5: TQ \mapsto \text{Target } TQ'$  and  $B6: (TP', TQ') \in (\text{TRel}^+)^{-1}$

by *blast*

from  $B2 B3$  trel have  $(TP', TR') \in \text{TRel}^+$

by *simp*

with  $B6$  have  $(TR', TQ') \in (\text{TRel}^+)^{-1}$

by *simp*

with  $B5$  show  $\exists TQ'. TQ \mapsto \text{Target } TQ' \wedge (TR', TQ') \in (\text{TRel}^+)^{-1}$

by *blast*

qed

qed

lemma (in *encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev*:

fixes  $\text{TRel} :: ('procT \times 'procT) \text{ set}$

assumes *sim*: *strong-reduction-simulation*  $((\text{indRelRTPO } \text{TRel})^{-1})$  (*STCal Source Target*)

shows *strong-reduction-simulation*  $((\text{TRel}^+)^{-1})$  *Target*

using *sim* *indRelRTPO.target*[**where**  $\text{TRel}=\text{TRel}$ ] *indRelRTPO-to-TRel*(4)[**where**  $\text{TRel}=\text{TRel}$ ]

*rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev*[**where**

$\text{Rel}=\text{indRelRTPO } \text{TRel}$  and  $\text{TRel}=\text{TRel}$ ]

by *blast*

lemma (in *encoding*) *indRelLTPO-impl-TRel-is-strong-reduction-simulation-rev*:

fixes  $\text{TRel} :: ('procT \times 'procT) \text{ set}$

assumes *sim*: *strong-reduction-simulation*  $((\text{indRelLTPO } \text{TRel})^{-1})$  (*STCal Source Target*)

shows *strong-reduction-simulation*  $((\text{TRel}^+)^{-1})$  *Target*

using *sim* *indRelLTPO.target*[**where**  $\text{TRel}=\text{TRel}$ ] *indRelLTPO-to-TRel*(4)[**where**  $\text{TRel}=\text{TRel}$ ]

*rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev*[**where**

$\text{Rel}=\text{indRelLTPO } \text{TRel}$  and  $\text{TRel}=\text{TRel}$ ]

by *blast*

lemma (in *encoding*) *rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation*:

fixes  $\text{TRel} :: ('procT \times 'procT) \text{ set}$

and  $\text{Rel} :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$

assumes *sim*: *strong-reduction-simulation*  $\text{Rel}$  (*STCal Source Target*)

and *target*:  $\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$

and *trel*:  $\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^*$

shows *strong-reduction-simulation*  $(\text{TRel}^*)$  *Target*

proof *clarify*

fix  $TP TQ TP'$

assume  $(TP, TQ) \in \text{TRel}^*$  and  $TP \mapsto \text{Target } TP'$

thus  $\exists TQ'. TQ \mapsto \text{Target } TQ' \wedge (TP', TQ') \in \text{TRel}^*$

proof (*induct arbitrary: TP'*)

fix  $TP'$

assume  $TP \mapsto \text{Target } TP'$

moreover have  $(TP', TP') \in \text{TRel}^*$

by *simp*

ultimately show  $\exists TQ'. TP \mapsto \text{Target } TQ' \wedge (TP', TQ') \in \text{TRel}^*$

by *blast*

next

case (*step TQ TR TP'*)

assume  $TP \mapsto \text{Target } TP'$

and  $\bigwedge TP'. TP \mapsto \text{Target } TP' \implies \exists TQ'. TQ \mapsto \text{Target } TQ' \wedge (TP', TQ') \in \text{TRel}^*$

from *this* obtain  $TQ'$  where  $B1: TQ \mapsto \text{Target } TQ'$  and  $B2: (TP', TQ') \in \text{TRel}^*$

by *blast*

assume  $(TQ, TR) \in \text{TRel}$



**with target have** (*TargetTerm*  $TQ$ , *TargetTerm*  $TR$ )  $\in$  *Rel*  
**by** *simp*  
**moreover from**  $B1$  **have** *TargetTerm*  $TQ \mapsto$  (*STCal* *Source* *Target*) (*TargetTerm*  $TQ'$ )  
**by** (*simp* *add*: *STCal-step*)  
**ultimately obtain**  $R'$  **where**  $B3$ : *TargetTerm*  $TR \mapsto$  (*STCal* *Source* *Target*)  $R'$   
**and**  $B4$ : (*TargetTerm*  $TQ'$ ,  $R'$ )  $\in$  *Rel*  
**using** *sim*  
**by** *blast*  
**from**  $B3$  **obtain**  $TR'$  **where**  $B5$ :  $TR' \in T R'$  **and**  $B6$ :  $TR \mapsto$  *Target*  $TR'$   
**by** (*auto simp* *add*: *STCal-step*)  
**from**  $B4 B5$  *tr* **have** ( $TQ'$ ,  $TR'$ )  $\in$  *TRel\**  
**by** *simp*  
**with**  $B2$  **have** ( $TP'$ ,  $TR'$ )  $\in$  *TRel\**  
**by** *simp*  
**with**  $B6$  **show**  $\exists TR'. TR \mapsto$  *Target*  $TR' \wedge$  ( $TP'$ ,  $TR'$ )  $\in$  *TRel\**  
**by** *blast*  
**qed**  
**qed**

**lemma** (**in** *encoding*) *indRelTEQ-impl-TRel-is-strong-reduction-simulation*:  
**fixes** *TRel* :: ('*procT*  $\times$  '*procT*) *set*  
**assumes** *sim*: *strong-reduction-simulation* (*indRelTEQ* *TRel*) (*STCal* *Source* *Target*)  
**shows** *strong-reduction-simulation* (*TRel\**) *Target*  
**using** *sim* *indRelTEQ.target* [**where**  $TRel = TRel$ ] *indRelTEQ-to-TRel(4)* [**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*  
*rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation* [**where**  
 $Rel = indRelTEQ TRel$  **and**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (**in** *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-simulation*:  
**fixes** *TRel* :: ('*procT*  $\times$  '*procT*) *set*  
**assumes** *sim*: *weak-barbed-simulation* (*indRelRTPO* *TRel*) (*STCalWB* *SWB* *TWB*)  
**shows** *weak-barbed-simulation* (*TRel*<sup>+</sup>) *TWB*  
**proof**  
**from** *sim* **show** *weak-reduction-simulation* (*TRel*<sup>+</sup>) (*Calculus* *TWB*)  
**using** *indRelRTPO-impl-TRel-is-weak-reduction-simulation* [**where**  $TRel = TRel$ ]  
**by** (*simp* *add*: *STCalWB-def* *calS* *calT*)  
**next**  
**from** *sim* **show** *rel-weakly-preserves-barbs* (*TRel*<sup>+</sup>) *TWB*  
**using** *indRelRTPO-impl-TRel-weakly-preserves-barbs* [**where**  $TRel = TRel$ ]  
*weak-preservation-of-barbs-and-closures(2)* [**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-simulation*:  
**fixes** *TRel* :: ('*procT*  $\times$  '*procT*) *set*  
**assumes** *sim*: *weak-barbed-simulation* (*indRelLTPO* *TRel*) (*STCalWB* *SWB* *TWB*)  
**shows** *weak-barbed-simulation* (*TRel*<sup>+</sup>) *TWB*  
**proof**  
**from** *sim* **show** *weak-reduction-simulation* (*TRel*<sup>+</sup>) (*Calculus* *TWB*)  
**using** *indRelLTPO-impl-TRel-is-weak-reduction-simulation* [**where**  $TRel = TRel$ ]  
**by** (*simp* *add*: *STCalWB-def* *calS* *calT*)  
**next**  
**from** *sim* **show** *rel-weakly-preserves-barbs* (*TRel*<sup>+</sup>) *TWB*  
**using** *indRelLTPO-impl-TRel-weakly-preserves-barbs* [**where**  $TRel = TRel$ ]  
*weak-preservation-of-barbs-and-closures(2)* [**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-simulation*:  
**fixes** *TRel* :: ('*procT*  $\times$  '*procT*) *set*

**assumes** *sim*: weak-barbed-simulation (indRelTEQ TRel) (STCalWB SWB TWB)  
**shows** weak-barbed-simulation (TRel\*) TWB  
**proof**  
**from** *sim* **show** weak-reduction-simulation (TRel\*) (Calculus TWB)  
**using** indRelTEQ-impl-TRel-is-weak-reduction-simulation[**where** TRel=TRel]  
**by** (simp add: STCalWB-def calS calT)  
**next**  
**from** *sim* **show** rel-weakly-preserves-barbs (TRel\*) TWB  
**using** indRelTEQ-impl-TRel-weakly-preserves-barbs[**where** TRel=TRel]  
weak-preservation-of-barbs-and-closures(3)[**where** Rel=TRel **and** CWB=TWB]  
**by** blast  
**qed**

**lemma** (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-strong-barbed-simulation:  
**fixes** TRel :: ('procT × 'procT) set  
**assumes** *sim*: strong-barbed-simulation (indRelRTPO TRel) (STCalWB SWB TWB)  
**shows** strong-barbed-simulation (TRel<sup>+</sup>) TWB  
**proof**  
**from** *sim* **show** strong-reduction-simulation (TRel<sup>+</sup>) (Calculus TWB)  
**using** indRelRTPO-impl-TRel-is-strong-reduction-simulation[**where** TRel=TRel]  
**by** (simp add: STCalWB-def calS calT)  
**next**  
**from** *sim* **show** rel-preserves-barbs (TRel<sup>+</sup>) TWB  
**using** indRelRTPO-impl-TRel-preserves-barbs[**where** TRel=TRel]  
preservation-of-barbs-and-closures(2)[**where** Rel=TRel **and** CWB=TWB]  
**by** blast  
**qed**

**lemma** (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-strong-barbed-simulation:  
**fixes** TRel :: ('procT × 'procT) set  
**assumes** *sim*: strong-barbed-simulation (indRelLTPO TRel) (STCalWB SWB TWB)  
**shows** strong-barbed-simulation (TRel<sup>+</sup>) TWB  
**proof**  
**from** *sim* **refl** **show** strong-reduction-simulation (TRel<sup>+</sup>) (Calculus TWB)  
**using** indRelLTPO-impl-TRel-is-strong-reduction-simulation[**where** TRel=TRel]  
**by** (simp add: STCalWB-def calS calT)  
**next**  
**from** *sim* **show** rel-preserves-barbs (TRel<sup>+</sup>) TWB  
**using** indRelLTPO-impl-TRel-preserves-barbs[**where** TRel=TRel]  
preservation-of-barbs-and-closures(2)[**where** Rel=TRel **and** CWB=TWB]  
**by** blast  
**qed**

**lemma** (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-strong-barbed-simulation:  
**fixes** TRel :: ('procT × 'procT) set  
**assumes** *sim*: strong-barbed-simulation (indRelTEQ TRel) (STCalWB SWB TWB)  
**shows** strong-barbed-simulation (TRel\*) TWB  
**proof**  
**from** *sim* **refl** **show** strong-reduction-simulation (TRel\*) (Calculus TWB)  
**using** indRelTEQ-impl-TRel-is-strong-reduction-simulation[**where** TRel=TRel]  
**by** (simp add: STCalWB-def calS calT)  
**next**  
**from** *sim* **show** rel-preserves-barbs (TRel\*) TWB  
**using** indRelTEQ-impl-TRel-preserves-barbs[**where** TRel=TRel]  
preservation-of-barbs-and-closures(3)[**where** Rel=TRel **and** CWB=TWB]  
**by** blast  
**qed**

If indRelRTPO, indRelLTPO, or indRelTEQ is a contrasimulation then so is the corresponding target term relation.

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$   
**assumes** *conSim*: *weak-reduction-contrasimulation Rel (STCal Source Target)*  
**and** *target*:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**and** *trel*:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**shows** *weak-reduction-contrasimulation (TRel<sup>+</sup>) Target*  
**proof** *clarify*  
**fix**  $TP TQ TP'$   
**assume**  $(TP, TQ) \in TRel^+$  **and**  $TP \mapsto Target^* TP'$   
**thus**  $\exists TQ'. TQ \mapsto Target^* TQ' \wedge (TQ', TP') \in TRel^+$   
**proof** (*induct arbitrary: TP'*)  
**fix**  $TQ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with** *target* **have**  $(TargetTerm TP, TargetTerm TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TP \mapsto Target^* TP'$   
**hence**  $TargetTerm TP \mapsto (STCal Source Target)^* (TargetTerm TP')$   
**by** (*simp add: STCal-steps*)  
**ultimately** **obtain**  $Q'$  **where**  $A2: TargetTerm TQ \mapsto (STCal Source Target)^* Q'$   
**and**  $A3: (Q', TargetTerm TP') \in Rel$   
**using** *conSim*  
**by** *blast*  
**from**  $A2$  **obtain**  $TQ'$  **where**  $A4: TQ \mapsto Target^* TQ'$  **and**  $A5: TQ' \in T Q'$   
**by** (*auto simp add: STCal-steps*)  
**from**  $A3 A5$  *trel* **have**  $(TQ', TP') \in TRel^+$   
**by** *simp*  
**with**  $A4$  **show**  $\exists TQ'. TQ \mapsto Target^* TQ' \wedge (TQ', TP') \in TRel^+$   
**by** *blast*  
**next**  
**case** (*step TQ TR*)  
**assume**  $TP \mapsto Target^* TP'$   
**and**  $\bigwedge TP'. TP \mapsto Target^* TP' \implies \exists TQ'. TQ \mapsto Target^* TQ' \wedge (TQ', TP') \in TRel^+$   
**from** *this* **obtain**  $TQ'$  **where**  $B1: TQ \mapsto Target^* TQ'$  **and**  $B2: (TQ', TP') \in TRel^+$   
**by** *blast*  
**assume**  $(TQ, TR) \in TRel$   
**with** *target* **have**  $(TargetTerm TQ, TargetTerm TR) \in Rel$   
**by** *simp*  
**moreover** **from**  $B1$  **have**  $TargetTerm TQ \mapsto (STCal Source Target)^* (TargetTerm TQ')$   
**by** (*simp add: STCal-steps*)  
**ultimately** **obtain**  $R'$  **where**  $B3: TargetTerm TR \mapsto (STCal Source Target)^* R'$   
**and**  $B4: (R', TargetTerm TQ') \in Rel$   
**using** *conSim*  
**by** *blast*  
**from**  $B3$  **obtain**  $TR'$  **where**  $B5: TR' \in T R'$  **and**  $B6: TR \mapsto Target^* TR'$   
**by** (*auto simp add: STCal-steps*)  
**from**  $B4 B5$  *trel* **have**  $(TR', TQ') \in TRel^+$   
**by** *simp*  
**from** *this*  $B2$  **have**  $(TR', TP') \in TRel^+$   
**by** *simp*  
**with**  $B6$  **show**  $\exists TR'. TR \mapsto Target^* TR' \wedge (TR', TP') \in TRel^+$   
**by** *blast*  
**qed**  
**qed**

**lemma** (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *conSim*: *weak-reduction-contrasimulation (indRelRTPO TRel) (STCal Source Target)*  
**shows** *weak-reduction-contrasimulation (TRel<sup>+</sup>) Target*  
**using** *conSim indRelRTPO.target* [**where**  $TRel = TRel$ ] *indRelRTPO-to-TRel(4)* [**where**  $TRel = TRel$ ]  
*rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation* [**where**

$Rel = indRelRTPO \ TRel \ \mathbf{and} \ TRel = TRel]$   
**by** *blast*

**lemma** (*in encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \ set$   
**assumes** *conSim*: *weak-reduction-contrasimulation* (*indRelLTPO*  $TRel$ ) (*STCal* *Source* *Target*)  
**shows** *weak-reduction-contrasimulation* ( $TRel^+$ ) *Target*  
**using** *conSim* *indRelLTPO.target*[**where**  $TRel = TRel$ ] *indRelLTPO-to-TRel*(4)[**where**  $TRel = TRel$ ]  
*rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation*[**where**  
 $Rel = indRelLTPO \ TRel \ \mathbf{and} \ TRel = TRel]$   
**by** *blast*

**lemma** (*in encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-contrasimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \ set$   
**and**  $Rel :: (('procS, 'procT) \ Proc \times ('procS, 'procT) \ Proc) \ set$   
**assumes** *conSim*: *weak-reduction-contrasimulation*  $Rel$  (*STCal* *Source* *Target*)  
**and** *target*:  $\forall T1 \ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in Rel$   
**and** *trel*:  $\forall T1 \ T2. (TargetTerm \ T1, TargetTerm \ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$   
**shows** *weak-reduction-contrasimulation* ( $TRel^*$ ) *Target*

**proof** *clarify*  
**fix**  $TP \ TQ \ TP'$   
**assume**  $(TP, TQ) \in TRel^*$  **and**  $TP \longmapsto Target^* \ TP'$   
**thus**  $\exists TQ'. TQ \longmapsto Target^* \ TQ' \wedge (TQ', TP') \in TRel^*$   
**proof** (*induct arbitrary*:  $TP'$ )  
**fix**  $TP'$   
**assume**  $TP \longmapsto Target^* \ TP'$   
**moreover** **have**  $(TP', TP') \in TRel^*$   
**by** *simp*  
**ultimately** **show**  $\exists TQ'. TP \longmapsto Target^* \ TQ' \wedge (TQ', TP') \in TRel^*$   
**by** *blast*

**next**  
**case** (*step*  $TQ \ TR$ )  
**assume**  $TP \longmapsto Target^* \ TP'$   
**and**  $\bigwedge TP'. TP \longmapsto Target^* \ TP' \implies \exists TQ'. TQ \longmapsto Target^* \ TQ' \wedge (TQ', TP') \in TRel^*$   
**from** *this* **obtain**  $TQ'$  **where**  $B1: TQ \longmapsto Target^* \ TQ'$  **and**  $B2: (TQ', TP') \in TRel^*$   
**by** *blast*  
**assume**  $(TQ, TR) \in TRel$   
**with** *target* **have**  $(TargetTerm \ TQ, TargetTerm \ TR) \in Rel$   
**by** *simp*  
**moreover** **from**  $B1$  **have**  $TargetTerm \ TQ \longmapsto (STCal \ Source \ Target)^* (TargetTerm \ TQ')$   
**by** (*simp* *add*: *STCal-steps*)  
**ultimately** **obtain**  $R'$  **where**  $B3: TargetTerm \ TR \longmapsto (STCal \ Source \ Target)^* \ R'$   
**and**  $B4: (R', TargetTerm \ TQ') \in Rel$   
**using** *conSim*  
**by** *blast*  
**from**  $B3$  **obtain**  $TR'$  **where**  $B5: TR' \in T \ R'$  **and**  $B6: TR \longmapsto Target^* \ TR'$   
**by** (*auto* *simp* *add*: *STCal-steps*)  
**from**  $B4 \ B5$  **trel** **have**  $(TR', TQ') \in TRel^*$   
**by** *simp*  
**from** *this*  $B2$  **have**  $(TR', TP') \in TRel^*$   
**by** *simp*  
**with**  $B6$  **show**  $\exists TR'. TR \longmapsto Target^* \ TR' \wedge (TR', TP') \in TRel^*$   
**by** *blast*

**qed**

**qed**

**lemma** (*in encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \ set$   
**assumes** *conSim*: *weak-reduction-contrasimulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
**shows** *weak-reduction-contrasimulation* ( $TRel^*$ ) *Target*  
**using** *conSim* *indRelTEQ.target*[**where**  $TRel = TRel$ ] *indRelTEQ-to-TRel*(4)[**where**  $TRel = TRel$ ]

*trans-closure-of-TRel-refl-cond*  
*rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-contrasimulation*[**where**  
*Rel=indRelTEQ TRel and TRel=TRel*]

by *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation*:  
**fixes** *TRel* :: ('procT × 'procT) set  
**assumes** *conSim*: *weak-barbed-contrasimulation* (*indRelRTPO TRel*) (*STCalWB SWB TWB*)  
**shows** *weak-barbed-contrasimulation* (*TRel*<sup>+</sup>) *TWB*  
**proof**  
**from** *conSim* **show** *weak-reduction-contrasimulation* (*TRel*<sup>+</sup>) (*Calculus TWB*)  
**using** *indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation*[**where** *TRel=TRel*]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *conSim* **show** *rel-weakly-preserves-barbs* (*TRel*<sup>+</sup>) *TWB*  
**using** *indRelRTPO-impl-TRel-weakly-preserves-barbs*[**where** *TRel=TRel*]  
*weak-preservation-of-barbs-and-closures*(2)[**where** *Rel=TRel and CWB=TWB*]  
**by** *blast*  
**qed**

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation*:  
**fixes** *TRel* :: ('procT × 'procT) set  
**assumes** *conSim*: *weak-barbed-contrasimulation* (*indRelLTPO TRel*) (*STCalWB SWB TWB*)  
**shows** *weak-barbed-contrasimulation* (*TRel*<sup>+</sup>) *TWB*  
**proof**  
**from** *conSim* **show** *weak-reduction-contrasimulation* (*TRel*<sup>+</sup>) (*Calculus TWB*)  
**using** *indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation*[**where** *TRel=TRel*]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *conSim* **show** *rel-weakly-preserves-barbs* (*TRel*<sup>+</sup>) *TWB*  
**using** *indRelLTPO-impl-TRel-weakly-preserves-barbs*[**where** *TRel=TRel*]  
*weak-preservation-of-barbs-and-closures*(2)[**where** *Rel=TRel and CWB=TWB*]  
**by** *blast*  
**qed**

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation*:  
**fixes** *TRel* :: ('procT × 'procT) set  
**assumes** *conSim*: *weak-barbed-contrasimulation* (*indRelTEQ TRel*) (*STCalWB SWB TWB*)  
**shows** *weak-barbed-contrasimulation* (*TRel*<sup>\*</sup>) *TWB*  
**proof**  
**from** *conSim* **show** *weak-reduction-contrasimulation* (*TRel*<sup>\*</sup>) (*Calculus TWB*)  
**using** *indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation*[**where** *TRel=TRel*]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *conSim* **show** *rel-weakly-preserves-barbs* (*TRel*<sup>\*</sup>) *TWB*  
**using** *indRelTEQ-impl-TRel-weakly-preserves-barbs*[**where** *TRel=TRel*]  
*weak-preservation-of-barbs-and-closures*(3)[**where** *Rel=TRel and CWB=TWB*]  
**by** *blast*  
**qed**

If *indRelRTPO*, *indRelLTPO*, or *indRelTEQ* is a coupled simulation then so is the corresponding target term relation.

**lemma** (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-coupled-simulation*:  
**fixes** *TRel* :: ('procT × 'procT) set  
**assumes** *couSim*: *weak-reduction-coupled-simulation* (*indRelRTPO TRel*) (*STCal Source Target*)  
**shows** *weak-reduction-coupled-simulation* (*TRel*<sup>+</sup>) *Target*  
**using** *couSim weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl indRelRTPO-impl-TRel-is-weak-reduction-simulation*[**where** *TRel=TRel*]  
*indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation*[**where** *TRel=TRel*]  
**by** *blast*

**lemma** (in *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *couSim*: *weak-reduction-coupled-simulation* (*indRelLTPO*  $TRel$ ) (*STCal* *Source* *Target*)  
**shows** *weak-reduction-coupled-simulation* ( $TRel^+$ ) *Target*  
**using** *couSim* *weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl* *indRelLTPO-impl-TRel-is-weak-reduction-simulation*[**where**  $TRel = TRel$ ]  
*indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (in *encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-coupled-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *couSim*: *weak-reduction-coupled-simulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
**shows** *weak-reduction-coupled-simulation* ( $TRel^*$ ) *Target*  
**using** *couSim* *weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl* *indRelTEQ-impl-TRel-is-weak-reduction-simulation*[**where**  $TRel = TRel$ ]  
*indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-coupled-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *couSim*: *weak-barbed-coupled-simulation* (*indRelRTPO*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
**shows** *weak-barbed-coupled-simulation* ( $TRel^+$ ) *TWB*  
**using** *couSim* *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl* *indRelRTPO-impl-TRel-is-weak-barbed-simulation*[**where**  $TRel = TRel$ ]  
*indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-coupled-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *couSim*: *weak-barbed-coupled-simulation* (*indRelLTPO*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
**shows** *weak-barbed-coupled-simulation* ( $TRel^+$ ) *TWB*  
**using** *couSim* *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl* *indRelLTPO-impl-TRel-is-weak-barbed-simulation*[**where**  $TRel = TRel$ ]  
*indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-coupled-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *couSim*: *weak-barbed-coupled-simulation* (*indRelTEQ*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
**shows** *weak-barbed-coupled-simulation* ( $TRel^*$ ) *TWB*  
**using** *couSim* *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation*  
*refl* *indRelTEQ-impl-TRel-is-weak-barbed-simulation*[**where**  $TRel = TRel$ ]  
*indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

If *indRelRTPO*, *indRelLTPO*, or *indRelTEQ* is a correspondence simulation then so is the corresponding target term relation.

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *corSim*: *weak-reduction-correspondence-simulation* *Rel* (*STCal* *Source* *Target*)  
**and** *target*:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**and** *trel*:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**shows** *weak-reduction-correspondence-simulation* ( $TRel^+$ ) *Target*  
**proof** –  
**from** *corSim* *target* *trel* **have** *A*: *weak-reduction-simulation* ( $TRel^+$ ) *Target*  
**using** *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation*[**where**  $TRel = TRel$ ]  
**and**  $Rel = Rel$   
**by** *blast*

**moreover have**  $\forall P Q Q'. (P, Q) \in TRel^+ \wedge Q \mapsto Target^* Q' \rightarrow (\exists P'' Q''. P \mapsto Target^* P'' \wedge Q' \mapsto Target^* Q'' \wedge (P'', Q'') \in TRel^+)$

**proof clarify**

**fix**  $TP TQ TQ'$

**assume**  $(TP, TQ) \in TRel^+$  **and**  $TQ \mapsto Target^* TQ'$

**thus**  $\exists TP'' TQ''. TP \mapsto Target^* TP'' \wedge TQ' \mapsto Target^* TQ'' \wedge (TP'', TQ'') \in TRel^+$

**proof** (*induct arbitrary: TQ'*)

**fix**  $TQ TQ'$

**assume**  $(TP, TQ) \in TRel$

**with target have**  $(TargetTerm TP, TargetTerm TQ) \in Rel$

**by** *blast*

**moreover assume**  $TQ \mapsto Target^* TQ'$

**hence**  $TargetTerm TQ \mapsto (STCal Source Target)^* (TargetTerm TQ')$

**by** (*simp add: STCal-steps*)

**ultimately obtain**  $P'' Q''$  **where**  $A2: TargetTerm TP \mapsto (STCal Source Target)^* P''$

**and**  $A3: TargetTerm TQ' \mapsto (STCal Source Target)^* Q''$  **and**  $A4: (P'', Q'') \in Rel$

**using** *corSim*

**by** *blast*

**from**  $A2$  **obtain**  $TP''$  **where**  $A5: TP \mapsto Target^* TP''$  **and**  $A6: TP'' \in T P''$

**by** (*auto simp add: STCal-steps*)

**from**  $A3$  **obtain**  $TQ''$  **where**  $A7: TQ' \mapsto Target^* TQ''$  **and**  $A8: TQ'' \in T Q''$

**by** (*auto simp add: STCal-steps*)

**from**  $A4 A6 A8$  **trel have**  $(TP'', TQ'') \in TRel^+$

**by** *blast*

**with**  $A5 A7$

**show**  $\exists TP'' TQ''. TP \mapsto Target^* TP'' \wedge TQ' \mapsto Target^* TQ'' \wedge (TP'', TQ'') \in TRel^+$

**by** *blast*

**next**

**case** (*step TQ TR TR'*)

**assume**  $\bigwedge TQ'. TQ \mapsto Target^* TQ' \implies \exists TP'' TQ''. TP \mapsto Target^* TP'' \wedge TQ' \mapsto Target^* TQ'' \wedge (TP'', TQ'') \in TRel^+$

**moreover assume**  $(TQ, TR) \in TRel$

**hence**  $\bigwedge TR'. TR \mapsto Target^* TR'$

$\rightarrow (\exists TQ'' TR''. TQ \mapsto Target^* TQ'' \wedge TR' \mapsto Target^* TR'' \wedge (TQ'', TR'') \in TRel^+)$

**proof clarify**

**fix**  $TR'$

**assume**  $(TQ, TR) \in TRel$

**with target have**  $(TargetTerm TQ, TargetTerm TR) \in Rel$

**by** *simp*

**moreover assume**  $TR \mapsto Target^* TR'$

**hence**  $TargetTerm TR \mapsto (STCal Source Target)^* (TargetTerm TR')$

**by** (*simp add: STCal-steps*)

**ultimately obtain**  $Q'' R''$  **where**  $B1: TargetTerm TQ \mapsto (STCal Source Target)^* Q''$

**and**  $B2: TargetTerm TR' \mapsto (STCal Source Target)^* R''$  **and**  $B3: (Q'', R'') \in Rel$

**using** *corSim*

**by** *blast*

**from**  $B1$  **obtain**  $TQ''$  **where**  $B4: TQ'' \in T Q''$  **and**  $B5: TQ \mapsto Target^* TQ''$

**by** (*auto simp add: STCal-steps*)

**from**  $B2$  **obtain**  $TR''$  **where**  $B6: TR'' \in T R''$  **and**  $B7: TR' \mapsto Target^* TR''$

**by** (*auto simp add: STCal-steps*)

**from**  $B3 B4 B6$  **trel have**  $(TQ'', TR'') \in TRel^+$

**by** *simp*

**with**  $B5 B7$

**show**  $\exists TQ'' TR''. TQ \mapsto Target^* TQ'' \wedge TR' \mapsto Target^* TR'' \wedge (TQ'', TR'') \in TRel^+$

**by** *blast*

**qed**

**moreover have** *trans*  $(TRel^+)$

**by** *simp*

**moreover assume**  $TR \mapsto Target^* TR'$

**ultimately**

**show**  $\exists TP'' TR''. TP \mapsto Target^* TP'' \wedge TR' \mapsto Target^* TR'' \wedge (TP'', TR'') \in TRel^+$

```

    using A reduction-correspondence-simulation-condition-trans[where Rel=TRel+
      and Cal=Target]
    by blast
  qed
qed
ultimately show ?thesis
  by simp
qed

```

**lemma** (in encoding) *indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation:*  
**fixes** TRel :: ('procT × 'procT) set  
**assumes** cSim: weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)  
**shows** weak-reduction-correspondence-simulation (TRel<sup>+</sup>) Target  
 using cSim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel(4)[where TRel=TRel]  
 rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where  
 Rel=indRelRTPO TRel and TRel=TRel]  
 by blast

**lemma** (in encoding) *indRelLTPO-impl-TRel-is-weak-reduction-correspondence-simulation:*  
**fixes** TRel :: ('procT × 'procT) set  
**assumes** cSim: weak-reduction-correspondence-simulation (indRelLTPO TRel) (STCal Source Target)  
**shows** weak-reduction-correspondence-simulation (TRel<sup>+</sup>) Target  
 using cSim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(4)[where TRel=TRel]  
 rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where  
 Rel=indRelLTPO TRel and TRel=TRel]  
 by blast

**lemma** (in encoding)  
*rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-correspondence-simulation:*  
**fixes** TRel :: ('procT × 'procT) set  
 and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set  
**assumes** corSim: weak-reduction-correspondence-simulation Rel (STCal Source Target)  
 and target:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
 and trel:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$   
**shows** weak-reduction-correspondence-simulation (TRel\*) Target  
**proof** –  
**from** corSim target trel **have** A: weak-reduction-simulation (TRel\*) Target  
 using rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation[where TRel=TRel  
 and Rel=Rel]  
 by blast  
**moreover have**  $\forall P Q Q'. (P, Q) \in TRel^* \wedge Q \longmapsto Target^* Q' \longrightarrow$   
 $(\exists P'' Q''. P \longmapsto Target^* P'' \wedge Q' \longmapsto Target^* Q'' \wedge (P'', Q'') \in TRel^*)$   
**proof** clarify  
**fix** TP TQ TQ'  
**assume** (TP, TQ)  $\in TRel^*$  and TQ  $\longmapsto Target^* TQ'$   
**thus**  $\exists TP'' TQ''. TP \longmapsto Target^* TP'' \wedge TQ' \longmapsto Target^* TQ'' \wedge (TP'', TQ'') \in TRel^*$   
**proof** (induct arbitrary: TQ')  
**fix** TQ'  
**assume** TP  $\longmapsto Target^* TQ'$   
**moreover have** TQ'  $\longmapsto Target^* TQ'$   
 by (simp add: steps-refl)  
**moreover have** (TQ', TQ')  $\in TRel^*$   
 by simp  
**ultimately show**  $\exists TP'' TQ''. TP \longmapsto Target^* TP'' \wedge TQ' \longmapsto Target^* TQ'' \wedge (TP'', TQ'') \in TRel^*$   
 by blast  
**next**  
**case** (step TQ TR TR')  
**assume**  $\bigwedge TQ'. TQ \longmapsto Target^* TQ' \implies \exists TP'' TQ''. TP \longmapsto Target^* TP'' \wedge TQ' \longmapsto Target^* TQ''$   
 $\wedge (TP'', TQ'') \in TRel^*$   
**moreover assume** (TQ, TR)  $\in TRel$   
**with** corSim **have**  $\bigwedge TR'. TR \longmapsto Target^* TR' \implies \exists TQ'' TR''. TQ \longmapsto Target^* TQ''$



$$\wedge TR' \mapsto \text{Target}^* TR'' \wedge (TQ'', TR'') \in T\text{Rel}^*$$

**proof** *clarify*

**fix**  $TR'$

**assume**  $(TQ, TR) \in T\text{Rel}$

**with** *target* **have**  $(\text{TargetTerm } TQ, \text{TargetTerm } TR) \in \text{Rel}$

**by** *simp*

**moreover** **assume**  $TR \mapsto \text{Target}^* TR'$

**hence**  $\text{TargetTerm } TR \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TR')$

**by**  $(\text{simp add: STCal-steps})$

**ultimately** **obtain**  $Q'' R''$  **where**  $B1: \text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* Q''$

**and**  $B2: \text{TargetTerm } TR' \mapsto (\text{STCal Source Target})^* R''$  **and**  $B3: (Q'', R'') \in \text{Rel}$

**using** *corSim*

**by** *blast*

**from**  $B1$  **obtain**  $TQ''$  **where**  $B4: TQ'' \in T Q''$  **and**  $B5: TQ \mapsto \text{Target}^* TQ''$

**by**  $(\text{auto simp add: STCal-steps})$

**from**  $B2$  **obtain**  $TR''$  **where**  $B6: TR'' \in T R''$  **and**  $B7: TR' \mapsto \text{Target}^* TR''$

**by**  $(\text{auto simp add: STCal-steps})$

**from**  $B3 B4 B6$  *trel* **have**  $(TQ'', TR'') \in T\text{Rel}^*$

**by** *simp*

**with**  $B5 B7$

**show**  $\exists TQ'' TR''. TQ \mapsto \text{Target}^* TQ'' \wedge TR' \mapsto \text{Target}^* TR'' \wedge (TQ'', TR'') \in T\text{Rel}^*$

**by** *blast*

**qed**

**moreover** **assume**  $TR \mapsto \text{Target}^* TR'$

**moreover** **have** *trans*  $(T\text{Rel}^*)$

**using** *trans-rtrancl*[*of*  $T\text{Rel}$ ]

**by** *simp*

**ultimately** **show**  $\exists TP'' TR''. TP \mapsto \text{Target}^* TP'' \wedge TR' \mapsto \text{Target}^* TR'' \wedge (TP'', TR'') \in T\text{Rel}^*$

**using** *A reduction-correspondence-simulation-condition-trans*[**where**  $\text{Rel} = T\text{Rel}^*$

**and**  $\text{Cal} = \text{Target}$ ]

**by** *blast*

**qed**

**qed**

**ultimately** **show** *?thesis*

**by** *simp*

**qed**

**lemma** (*in encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation*:

**fixes**  $T\text{Rel} :: ('procT \times 'procT) \text{ set}$

**assumes** *corSim*: *weak-reduction-correspondence-simulation*  $(\text{indRelTEQ } T\text{Rel}) (\text{STCal Source Target})$

**shows** *weak-reduction-correspondence-simulation*  $(T\text{Rel}^*) \text{Target}$

**using** *corSim* *indRelTEQ.target*[**where**  $T\text{Rel} = T\text{Rel}$ ] *indRelTEQ-to-TRel*(4)[**where**  $T\text{Rel} = T\text{Rel}$ ]

*trans-closure-of-TRel-refl-cond*

*rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-correspondence-simulation*[

**where**  $\text{Rel} = \text{indRelTEQ } T\text{Rel}$  **and**  $T\text{Rel} = T\text{Rel}$ ]

**by** *blast*

**lemma** (*in encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-correspondence-simulation*:

**fixes**  $T\text{Rel} :: ('procT \times 'procT) \text{ set}$

**assumes** *corSim*: *weak-barbed-correspondence-simulation*  $(\text{indRelRTPO } T\text{Rel}) (\text{STCalWB SWB TWB})$

**shows** *weak-barbed-correspondence-simulation*  $(T\text{Rel}^+) \text{TWB}$

**proof**

**from** *corSim* **show** *weak-reduction-correspondence-simulation*  $(T\text{Rel}^+) (\text{Calculus TWB})$

**using** *indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation*[**where**  $T\text{Rel} = T\text{Rel}$ ]

**by**  $(\text{simp add: STCalWB-def calS calT})$

**next**

**from** *corSim* **show** *rel-weakly-respects-barbs*  $(T\text{Rel}^+) \text{TWB}$

**using** *indRelRTPO-impl-TRel-weakly-respects-barbs*[**where**  $T\text{Rel} = T\text{Rel}$ ]

*weak-respection-of-barbs-and-closures*(3)[**where**  $\text{Rel} = T\text{Rel}$  **and**  $\text{CWB} = \text{TWB}$ ]

**by** *blast*

**qed**

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-correspondence-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *corSim*: *weak-barbed-correspondence-simulation* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *weak-barbed-correspondence-simulation* ( $TRel^+$ )  $TWB$   
**proof**  
**from** *corSim* **show** *weak-reduction-correspondence-simulation* ( $TRel^+$ ) (*Calculus TWB*)  
**using** *indRelLTPO-impl-TRel-is-weak-reduction-correspondence-simulation*[**where**  $TRel = TRel$ ]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *corSim* **show** *rel-weakly-respects-barbs* ( $TRel^+$ )  $TWB$   
**using** *indRelLTPO-impl-TRel-weakly-respects-barbs*[**where**  $TRel = TRel$ ]  
*weak-respection-of-barbs-and-closures*(3)[**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-correspondence-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *corSim*: *weak-barbed-correspondence-simulation* (*indRelTEQ*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *weak-barbed-correspondence-simulation* ( $TRel^*$ )  $TWB$   
**proof**  
**from** *corSim* **show** *weak-reduction-correspondence-simulation* ( $TRel^*$ ) (*Calculus TWB*)  
**using** *indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation*[**where**  $TRel = TRel$ ]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *corSim* **show** *rel-weakly-respects-barbs* ( $TRel^*$ )  $TWB$   
**using** *indRelTEQ-impl-TRel-weakly-respects-barbs*[**where**  $TRel = TRel$ ]  
*weak-respection-of-barbs-and-closures*(5)[**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

If *indRelRTPO*, *indRelLTPO*, or *indRelTEQ* is a bisimulation then so is the corresponding target term relation.

**lemma** (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *bisim*: *weak-reduction-bisimulation*  $Rel$  (*STCal Source Target*)  
**and** *target*:  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**and** *trel*:  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**shows** *weak-reduction-bisimulation* ( $TRel^+$ )  $Target$   
**proof**  
**from** *bisim target trel* **show** *weak-reduction-simulation* ( $TRel^+$ )  $Target$   
**using** *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation*[**where**  $TRel = TRel$   
**and**  $Rel = Rel$ ]  
**by** *blast*  
**next**  
**show**  $\forall P Q Q'. (P, Q) \in TRel^+ \wedge Q \mapsto Target^* Q' \longrightarrow (\exists P'. P \mapsto Target^* P' \wedge (P', Q') \in TRel^+)$   
**proof** *clarify*  
**fix**  $TP TQ TQ'$   
**assume**  $(TP, TQ) \in TRel^+$  **and**  $TQ \mapsto Target^* TQ'$   
**thus**  $\exists TP'. TP \mapsto Target^* TP' \wedge (TP', TQ') \in TRel^+$   
**proof** (*induct arbitrary: TQ'*)  
**fix**  $TQ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with** *target* **have**  $(TargetTerm TP, TargetTerm TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TQ \mapsto Target^* TQ'$   
**hence**  $TargetTerm TQ \mapsto (STCal Source Target)^* (TargetTerm TQ')$   
**by** (*simp add: STCal-steps*)  
**ultimately obtain**  $P'$  **where**  $A2: TargetTerm TP \mapsto (STCal Source Target)^* P'$

**and**  $A3: (P', \text{TargetTerm } TQ') \in \text{Rel}$   
**using** *bisim*  
**by** *blast*  
**from**  $A2$  **obtain**  $TP'$  **where**  $A4: TP \mapsto \text{Target}^* TP'$  **and**  $A5: TP' \in T P'$   
**by** (*auto simp add: STCal-steps*)  
**from**  $A3$   $A5$  *trel* **have**  $(TP', TQ') \in \text{TRel}^+$   
**by** *simp*  
**with**  $A4$  **show**  $\exists TP'. TP \mapsto \text{Target}^* TP' \wedge (TP', TQ') \in \text{TRel}^+$   
**by** *blast*  
**next**  
**case** (*step TQ TR TR'*)  
**assume**  $(TQ, TR) \in \text{TRel}$   
**with** *target* **have**  $(\text{TargetTerm } TQ, \text{TargetTerm } TR) \in \text{Rel}$   
**by** *simp*  
**moreover** **assume**  $TR \mapsto \text{Target}^* TR'$   
**hence**  $\text{TargetTerm } TR \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TR')$   
**by** (*simp add: STCal-steps*)  
**ultimately** **obtain**  $Q'$  **where**  $B1: \text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* Q'$   
**and**  $B2: (Q', \text{TargetTerm } TR') \in \text{Rel}$   
**using** *bisim*  
**by** *blast*  
**from**  $B1$  **obtain**  $TQ'$  **where**  $B3: TQ' \in T Q'$  **and**  $B4: TQ \mapsto \text{Target}^* TQ'$   
**by** (*auto simp add: STCal-steps*)  
**assume**  $\bigwedge TQ'. TQ \mapsto \text{Target}^* TQ' \implies \exists TP'. TP \mapsto \text{Target}^* TP' \wedge (TP', TQ') \in \text{TRel}^+$   
**with**  $B4$  **obtain**  $TP'$  **where**  $B5: TP \mapsto \text{Target}^* TP'$  **and**  $B6: (TP', TQ') \in \text{TRel}^+$   
**by** *blast*  
**from**  $B2$   $B3$  *trel* **have**  $(TQ', TR') \in \text{TRel}^+$   
**by** *simp*  
**with**  $B6$  **have**  $(TP', TR') \in \text{TRel}^+$   
**by** *simp*  
**with**  $B5$  **show**  $\exists TP'. TP \mapsto \text{Target}^* TP' \wedge (TP', TR') \in \text{TRel}^+$   
**by** *blast*  
**qed**  
**qed**  
**qed**

**lemma** (*in encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-bisimulation:*

**fixes**  $\text{TRel} :: ('procT \times 'procT) \text{ set}$   
**assumes** *bisim: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)*  
**shows** *weak-reduction-bisimulation (TRel<sup>+</sup>) Target*  
**using** *bisim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel(4)[where TRel=TRel]*  
*rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation[where*  
*Rel=indRelRTPO TRel and TRel=TRel]*  
**by** *blast*

**lemma** (*in encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-bisimulation:*

**fixes**  $\text{TRel} :: ('procT \times 'procT) \text{ set}$   
**assumes** *bisim: weak-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)*  
**shows** *weak-reduction-bisimulation (TRel<sup>+</sup>) Target*  
**using** *bisim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(4)[where TRel=TRel]*  
*rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation[where*  
*Rel=indRelLTPO TRel and TRel=TRel]*  
**by** *blast*

**lemma** (*in encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation:*

**fixes**  $\text{TRel} :: ('procT \times 'procT) \text{ set}$   
**and**  $\text{Rel} :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *bisim: weak-reduction-bisimulation Rel (STCal Source Target)*  
**and** *target:  $\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$*   
**and** *trel:  $\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^*$*   
**shows** *weak-reduction-bisimulation (TRel<sup>\*</sup>) Target*

**proof**  
**from** *bisim target trel* **show** *weak-reduction-simulation* ( $TRel^*$ ) *Target*  
**using** *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation*[**where**  $TRel = TRel$   
**and**  $Rel = Rel$ ]  
**by** *blast*  
**next**  
**show**  $\forall P Q Q'. (P, Q) \in TRel^* \wedge Q \mapsto Target^* Q' \longrightarrow (\exists P'. P \mapsto Target^* P' \wedge (P', Q') \in TRel^*)$   
**proof** *clarify*  
**fix**  $TP\ TQ\ TQ'$   
**assume**  $(TP, TQ) \in TRel^*$  **and**  $TQ \mapsto Target^* TQ'$   
**thus**  $\exists TP'. TP \mapsto Target^* TP' \wedge (TP', TQ') \in TRel^*$   
**proof** (*induct arbitrary: TQ'*)  
**fix**  $TQ'$   
**assume**  $TP \mapsto Target^* TQ'$   
**moreover** **have**  $(TQ', TQ') \in TRel^*$   
**by** *simp*  
**ultimately** **show**  $\exists TP'. TP \mapsto Target^* TP' \wedge (TP', TQ') \in TRel^*$   
**by** *blast*  
**next**  
**case** (*step TQ TR TR'*)  
**assume**  $(TQ, TR) \in TRel$   
**with** *target* **have**  $(TargetTerm\ TQ, TargetTerm\ TR) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TR \mapsto Target^* TR'$   
**hence**  $TargetTerm\ TR \mapsto (STCal\ Source\ Target)^* (TargetTerm\ TR')$   
**by** (*simp add: STCal-steps*)  
**ultimately** **obtain**  $Q'$  **where**  $B1: TargetTerm\ TQ \mapsto (STCal\ Source\ Target)^* Q'$   
**and**  $B2: (Q', TargetTerm\ TR') \in Rel$   
**using** *bisim*  
**by** *blast*  
**from**  $B1$  **obtain**  $TQ'$  **where**  $B3: TQ' \in T\ Q'$  **and**  $B4: TQ \mapsto Target^* TQ'$   
**by** (*auto simp add: STCal-steps*)  
**assume**  $\bigwedge TQ'. TQ \mapsto Target^* TQ' \implies \exists TP'. TP \mapsto Target^* TP' \wedge (TP', TQ') \in TRel^*$   
**with**  $B4$  **obtain**  $TP'$  **where**  $B5: TP \mapsto Target^* TP'$  **and**  $B6: (TP', TQ') \in TRel^*$   
**by** *blast*  
**from**  $B2\ B3$  *trel* **have**  $(TQ', TR') \in TRel^*$   
**by** *simp*  
**with**  $B6$  **have**  $(TP', TR') \in TRel^*$   
**by** *simp*  
**with**  $B5$  **show**  $\exists TP'. TP \mapsto Target^* TP' \wedge (TP', TR') \in TRel^*$   
**by** *blast*  
**qed**  
**qed**  
**qed**

**lemma** (**in** *encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT)\ set$   
**assumes** *bisim: weak-reduction-bisimulation* (*indRelTEQ TRel*) (*STCal Source Target*)  
**shows** *weak-reduction-bisimulation* ( $TRel^*$ ) *Target*  
**using** *bisim indRelTEQ.target*[**where**  $TRel = TRel$ ] *indRelTEQ-to-TRel(4)*[**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*  
*rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation*[**where**  
 $Rel = indRelTEQ\ TRel$  **and**  $TRel = TRel$ ]  
**by** *blast*

**lemma** (**in** *encoding*) *rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT)\ set$   
**and**  $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$   
**assumes** *bisim: strong-reduction-bisimulation*  $Rel$  (*STCal Source Target*)  
**and** *target*:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and** *trel*:  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$

**shows** *strong-reduction-bisimulation* ( $TRel^+$ ) *Target*  
**proof**  
**from** *bisim target trel show strong-reduction-simulation* ( $TRel^+$ ) *Target*  
**using** *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation*[**where**  $Rel=Rel$   
**and**  $TRel=TRel$ ]  
**by** *blast*  
**next**  
**show**  $\forall P Q Q'. (P, Q) \in TRel^+ \wedge Q \mapsto Target Q' \longrightarrow (\exists P'. P \mapsto Target P' \wedge (P', Q') \in TRel^+)$   
**proof** *clarify*  
**fix**  $TP TQ TQ'$   
**assume**  $(TP, TQ) \in TRel^+$  **and**  $TQ \mapsto Target TQ'$   
**thus**  $\exists TP'. TP \mapsto Target TP' \wedge (TP', TQ') \in TRel^+$   
**proof** (*induct arbitrary: TQ'*)  
**fix**  $TQ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with** *target have*  $(TargetTerm TP, TargetTerm TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TQ \mapsto Target TQ'$   
**hence**  $TargetTerm TQ \mapsto (STCal Source Target) (TargetTerm TQ')$   
**by** (*simp add: STCal-step*)  
**ultimately obtain**  $P'$  **where**  $A2: TargetTerm TP \mapsto (STCal Source Target) P'$   
**and**  $A3: (P', TargetTerm TQ') \in Rel$   
**using** *bisim*  
**by** *blast*  
**from**  $A2$  **obtain**  $TP'$  **where**  $A4: TP \mapsto Target TP'$  **and**  $A5: TP' \in T P'$   
**by** (*auto simp add: STCal-step*)  
**from**  $A3 A5$  *trel have*  $(TP', TQ') \in TRel^+$   
**by** *simp*  
**with**  $A4$  **show**  $\exists TP'. TP \mapsto Target TP' \wedge (TP', TQ') \in TRel^+$   
**by** *blast*  
**next**  
**case** (*step TQ TR TR'*)  
**assume**  $(TQ, TR) \in TRel$   
**with** *target have*  $(TargetTerm TQ, TargetTerm TR) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TR \mapsto Target TR'$   
**hence**  $TargetTerm TR \mapsto (STCal Source Target) (TargetTerm TR')$   
**by** (*simp add: STCal-step*)  
**ultimately obtain**  $Q'$  **where**  $B1: TargetTerm TQ \mapsto (STCal Source Target) Q'$   
**and**  $B2: (Q', TargetTerm TR') \in Rel$   
**using** *bisim*  
**by** *blast*  
**from**  $B1$  **obtain**  $TQ'$  **where**  $B3: TQ' \in T Q'$  **and**  $B4: TQ \mapsto Target TQ'$   
**by** (*auto simp add: STCal-step*)  
**assume**  $\bigwedge TQ'. TQ \mapsto Target TQ' \implies \exists TP'. TP \mapsto Target TP' \wedge (TP', TQ') \in TRel^+$   
**with**  $B4$  **obtain**  $TP'$  **where**  $B5: TP \mapsto Target TP'$  **and**  $B6: (TP', TQ') \in TRel^+$   
**by** *blast*  
**from**  $B2 B3$  *trel have*  $(TQ', TR') \in TRel^+$   
**by** *simp*  
**with**  $B6$  **have**  $(TP', TR') \in TRel^+$   
**by** *simp*  
**with**  $B5$  **show**  $\exists TP'. TP \mapsto Target TP' \wedge (TP', TR') \in TRel^+$   
**by** *blast*  
**qed**  
**qed**  
**qed**

**lemma** (**in** *encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT) set$   
**assumes** *bisim: strong-reduction-bisimulation* (*indRelRTPO TRel*) (*STCal Source Target*)  
**shows** *strong-reduction-bisimulation* ( $TRel^+$ ) *Target*

using *bisim indRelRTPO.target*[**where**  $TRel = TRel$ ] *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]  
*rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation*[**where**  
 $Rel = indRelRTPO\ TRel$  **and**  $TRel = TRel$ ]  
by *blast*

**lemma** (in *encoding*) *indRelLTPO-impl-TRel-is-strong-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT)\ set$   
**assumes** *bisim: strong-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)*  
**shows** *strong-reduction-bisimulation (TRel<sup>+</sup>) Target*  
using *bisim indRelLTPO.target*[**where**  $TRel = TRel$ ] *indRelLTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]  
*rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation*[**where**  
 $Rel = indRelLTPO\ TRel$  **and**  $TRel = TRel$ ]  
by *blast*

**lemma** (in *encoding*) *rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT)\ set$   
**and**  $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$   
**assumes** *bisim: strong-reduction-bisimulation Rel (STCal Source Target)*  
**and** *target:  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$*   
**and** *trel:  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$*   
**shows** *strong-reduction-bisimulation (TRel<sup>\*</sup>) Target*  
**proof**  
**from** *bisim target trel* **show** *strong-reduction-simulation (TRel<sup>\*</sup>) Target*  
using *rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation*[**where**  $Rel = Rel$   
**and**  $TRel = TRel$ ]  
by *blast*

**next**  
**show**  $\forall P\ Q\ Q'. (P, Q) \in TRel^* \wedge Q \longmapsto Target\ Q' \longrightarrow (\exists P'. P \longmapsto Target\ P' \wedge (P', Q') \in TRel^*)$   
**proof** *clarify*  
**fix**  $TP\ TQ\ TQ'$   
**assume**  $(TP, TQ) \in TRel^*$  **and**  $TQ \longmapsto Target\ TQ'$   
**thus**  $\exists TP'. TP \longmapsto Target\ TP' \wedge (TP', TQ') \in TRel^*$   
**proof** (*induct arbitrary: TQ'*)  
**fix**  $TQ'$   
**assume**  $TP \longmapsto Target\ TQ'$   
**thus**  $\exists TP'. TP \longmapsto Target\ TP' \wedge (TP', TQ') \in TRel^*$   
by *blast*  
**next**  
**case** (*step TQ TR TR'*)  
**assume**  $(TQ, TR) \in TRel$   
**with** *target* **have**  $(TargetTerm\ TQ, TargetTerm\ TR) \in Rel$   
by *simp*  
**moreover** **assume**  $TR \longmapsto Target\ TR'$   
**hence**  $TargetTerm\ TR \longmapsto (STCal\ Source\ Target)\ (TargetTerm\ TR')$   
by (*simp add: STCal-step*)  
**ultimately obtain**  $Q'$  **where**  $B1: TargetTerm\ TQ \longmapsto (STCal\ Source\ Target)\ Q'$   
**and**  $B2: (Q', TargetTerm\ TR') \in Rel$   
using *bisim*  
by *blast*  
**from**  $B1$  **obtain**  $TQ'$  **where**  $B3: TQ' \in T\ Q'$  **and**  $B4: TQ \longmapsto Target\ TQ'$   
by (*auto simp add: STCal-step*)  
**assume**  $\bigwedge TQ'. TQ \longmapsto Target\ TQ' \implies \exists TP'. TP \longmapsto Target\ TP' \wedge (TP', TQ') \in TRel^*$   
**with**  $B4$  **obtain**  $TP'$  **where**  $B5: TP \longmapsto Target\ TP'$  **and**  $B6: (TP', TQ') \in TRel^*$   
by *blast*  
**from**  $B2\ B3\ trel$  **have**  $(TQ', TR') \in TRel^*$   
by *simp*  
**with**  $B6$  **have**  $(TP', TR') \in TRel^*$   
by *simp*  
**with**  $B5$  **show**  $\exists TP'. TP \longmapsto Target\ TP' \wedge (TP', TR') \in TRel^*$   
by *blast*  
**qed**

qed  
qed

**lemma** (in *encoding*) *indRelTEQ-impl-TRel-is-strong-reduction-bisimulation*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *bisim*: *strong-reduction-bisimulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
shows *strong-reduction-bisimulation* ( $TRel^*$ ) *Target*  
using *bisim* *indRelTEQ.target* [where  $TRel = TRel$ ] *indRelTEQ-to-TRel(4)* [where  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*  
*rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation* [where  
 $Rel = indRelTEQ\ TRel$  and  $TRel = TRel$ ]  
by *blast*

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-bisimulation*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *bisim*: *weak-barbed-bisimulation* (*indRelRTPO*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
shows *weak-barbed-bisimulation* ( $TRel^+$ ) *TWB*  
**proof**  
from *bisim* show *weak-reduction-bisimulation* ( $TRel^+$ ) (*Calculus* *TWB*)  
using *indRelRTPO-impl-TRel-is-weak-reduction-bisimulation* [where  $TRel = TRel$ ]  
by (*simp* *add*: *STCalWB-def* *calS* *calT*)  
next  
from *bisim* show *rel-weakly-respects-barbs* ( $TRel^+$ ) *TWB*  
using *indRelRTPO-impl-TRel-weakly-respects-barbs* [where  $TRel = TRel$ ]  
*weak-respection-of-barbs-and-closures(3)* [where  $Rel = TRel$  and  $CWB = TWB$ ]  
by *blast*  
qed

**lemma** (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-bisimulation*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *bisim*: *weak-barbed-bisimulation* (*indRelLTPO*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
shows *weak-barbed-bisimulation* ( $TRel^+$ ) *TWB*  
**proof**  
from *bisim* show *weak-reduction-bisimulation* ( $TRel^+$ ) (*Calculus* *TWB*)  
using *indRelLTPO-impl-TRel-is-weak-reduction-bisimulation* [where  $TRel = TRel$ ]  
by (*simp* *add*: *STCalWB-def* *calS* *calT*)  
next  
from *bisim* show *rel-weakly-respects-barbs* ( $TRel^+$ ) *TWB*  
using *indRelLTPO-impl-TRel-weakly-respects-barbs* [where  $TRel = TRel$ ]  
*weak-respection-of-barbs-and-closures(3)* [where  $Rel = TRel$  and  $CWB = TWB$ ]  
by *blast*  
qed

**lemma** (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-bisimulation*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *bisim*: *weak-barbed-bisimulation* (*indRelTEQ*  $TRel$ ) (*STCalWB* *SWB* *TWB*)  
shows *weak-barbed-bisimulation* ( $TRel^*$ ) *TWB*  
**proof**  
from *bisim* show *weak-reduction-bisimulation* ( $TRel^*$ ) (*Calculus* *TWB*)  
using *indRelTEQ-impl-TRel-is-weak-reduction-bisimulation* [where  $TRel = TRel$ ]  
by (*simp* *add*: *STCalWB-def* *calS* *calT*)  
next  
from *bisim* show *rel-weakly-respects-barbs* ( $TRel^*$ ) *TWB*  
using *indRelTEQ-impl-TRel-weakly-respects-barbs* [where  $TRel = TRel$ ]  
*weak-respection-of-barbs-and-closures(5)* [where  $Rel = TRel$  and  $CWB = TWB$ ]  
by *blast*  
qed

**lemma** (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-strong-barbed-bisimulation*:  
fixes  $TRel :: ('procT \times 'procT) \text{ set}$   
assumes *bisim*: *strong-barbed-bisimulation* (*indRelRTPO*  $TRel$ ) (*STCalWB* *SWB* *TWB*)

**shows** *strong-barbed-bisimulation* ( $TRel^+$ )  $TWB$   
**proof**  
**from** *bisim show strong-reduction-bisimulation* ( $TRel^+$ ) (*Calculus*  $TWB$ )  
**using** *indRelRTPO-impl-TRel-is-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *bisim show rel-respects-barbs* ( $TRel^+$ )  $TWB$   
**using** *indRelRTPO-impl-TRel-respects-barbs*[**where**  $TRel = TRel$ ]  
*respection-of-barbs-and-closures*(3)[**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-strong-barbed-bisimulation:*  
**fixes**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes** *bisim: strong-barbed-bisimulation* (*indRelLTPO*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *strong-barbed-bisimulation* ( $TRel^+$ )  $TWB$   
**proof**  
**from** *bisim refl show strong-reduction-bisimulation* ( $TRel^+$ ) (*Calculus*  $TWB$ )  
**using** *indRelLTPO-impl-TRel-is-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *bisim show rel-respects-barbs* ( $TRel^+$ )  $TWB$   
**using** *indRelLTPO-impl-TRel-respects-barbs*[**where**  $TRel = TRel$ ]  
*respection-of-barbs-and-closures*(3)[**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-strong-barbed-bisimulation:*  
**fixes**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes** *bisim: strong-barbed-bisimulation* (*indRelTEQ*  $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *strong-barbed-bisimulation* ( $TRel^*$ )  $TWB$   
**proof**  
**from** *bisim refl show strong-reduction-bisimulation* ( $TRel^*$ ) (*Calculus*  $TWB$ )  
**using** *indRelTEQ-impl-TRel-is-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** (*simp add: STCalWB-def calS calT*)  
**next**  
**from** *bisim show rel-respects-barbs* ( $TRel^*$ )  $TWB$   
**using** *indRelTEQ-impl-TRel-respects-barbs*[**where**  $TRel = TRel$ ]  
*respection-of-barbs-and-closures*(5)[**where**  $Rel = TRel$  **and**  $CWB = TWB$ ]  
**by** *blast*  
**qed**

### 5.3 Relations Induced by the Encoding and Relations on Source Terms and Target Terms

Some encodability like e.g. full abstraction are defined w.r.t. a relation on source terms and a relation on target terms. To analyse such criteria we include these two relations in the considered relation on the disjoint union of source and target terms.

**inductive-set** (**in** *encoding*) *indRelRST*  
 $:: ('procS \times 'procS)$  *set*  $\Rightarrow ('procT \times 'procT)$  *set*  
 $\Rightarrow (((('procS, 'procT)$  *Proc*)  $\times$  ( $'procS, 'procT$ ) *Proc*)) *set*  
**for**  $SRel :: ('procS \times 'procS)$  *set*  
**and**  $TRel :: ('procT \times 'procT)$  *set*  
**where**  
*encR:* (*SourceTerm*  $S$ , *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *indRelRST*  $SRel$   $TRel$  |  
*source:* ( $S1, S2$ )  $\in$   $SRel \Longrightarrow$  (*SourceTerm*  $S1$ , *SourceTerm*  $S2$ )  $\in$  *indRelRST*  $SRel$   $TRel$  |  
*target:* ( $T1, T2$ )  $\in$   $TRel \Longrightarrow$  (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$  *indRelRST*  $SRel$   $TRel$

**abbreviation** (**in** *encoding*) *indRelRSTinfix*  
 $:: ('procS, 'procT)$  *Proc*  $\Rightarrow ('procS \times 'procS)$  *set*  $\Rightarrow ('procT \times 'procT)$  *set*



$\Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \mathcal{R}[\cdot]R<-,> - [75, 75, 75, 75] 80)$

**where**

$P \mathcal{R}[\cdot]R<SRel, TRel> Q \equiv (P, Q) \in indRelRST\ SRel\ TRel$

**inductive-set** (in encoding) *indRelRSTPO*

$:: ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set$   
 $\Rightarrow (((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set$

**for**  $SRel :: ('procS \times 'procS) set$

**and**  $TRel :: ('procT \times 'procT) set$

**where**

*encR*:  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelRSTPO\ SRel\ TRel \mid$

*source*:  $(S1, S2) \in SRel \Longrightarrow (SourceTerm\ S1, SourceTerm\ S2) \in indRelRSTPO\ SRel\ TRel \mid$

*target*:  $(T1, T2) \in TRel \Longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in indRelRSTPO\ SRel\ TRel \mid$

*trans*:  $\llbracket (P, Q) \in indRelRSTPO\ SRel\ TRel; (Q, R) \in indRelRSTPO\ SRel\ TRel \rrbracket$   
 $\Longrightarrow (P, R) \in indRelRSTPO\ SRel\ TRel$

**abbreviation** (in encoding) *indRelRSTPOinfix* ::

$('procS, 'procT) Proc \Rightarrow ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set$

$\Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \lesssim[\cdot]R<-,> - [75, 75, 75, 75] 80)$

**where**

$P \lesssim[\cdot]R<SRel, TRel> Q \equiv (P, Q) \in indRelRSTPO\ SRel\ TRel$

**lemma** (in encoding) *indRelRSTPO-refl*:

**fixes**  $SRel :: ('procS \times 'procS) set$

**and**  $TRel :: ('procT \times 'procT) set$

**assumes** *reflS*: *refl*  $SRel$

**and** *reflT*: *refl*  $TRel$

**shows** *refl* (*indRelRSTPO*  $SRel\ TRel$ )

**unfolding** *refl-on-def*

**proof** *auto*

**fix**  $P$

**show**  $P \lesssim[\cdot]R<SRel, TRel> P$

**proof** (*cases*  $P$ )

**case** (*SourceTerm*  $SP$ )

**assume**  $SP \in S\ P$

**with** *reflS* **show**  $P \lesssim[\cdot]R<SRel, TRel> P$

**unfolding** *refl-on-def*

**by** (*simp* *add*: *indRelRSTPO.source*)

**next**

**case** (*TargetTerm*  $TP$ )

**assume**  $TP \in T\ P$

**with** *reflT* **show**  $P \lesssim[\cdot]R<SRel, TRel> P$

**unfolding** *refl-on-def*

**by** (*simp* *add*: *indRelRSTPO.target*)

**qed**

**qed**

**lemma** (in encoding) *indRelRSTPO-trans*:

**fixes**  $SRel :: ('procS \times 'procS) set$

**and**  $TRel :: ('procT \times 'procT) set$

**shows** *trans* (*indRelRSTPO*  $SRel\ TRel$ )

**unfolding** *trans-def*

**proof** *clarify*

**fix**  $P\ Q\ R$

**assume**  $P \lesssim[\cdot]R<SRel, TRel> Q$  **and**  $Q \lesssim[\cdot]R<SRel, TRel> R$

**thus**  $P \lesssim[\cdot]R<SRel, TRel> R$

**by** (*rule* *indRelRSTPO.trans*)

**qed**

**lemma** (in encoding) *refl-trans-closure-of-indRelRST*:

**fixes**  $SRel :: ('procS \times 'procS) set$

```

    and TRel :: ('procT × 'procT) set
  assumes reflS: refl SRel
    and reflT: refl TRel
  shows indRelRSTPO SRel TRel = (indRelRST SRel TRel)*
proof auto
  fix P Q
  assume P ≲[·]R<SRel,TRel> Q
  thus (P, Q) ∈ (indRelRST SRel TRel)*
proof induct
  case (encR S)
  show (SourceTerm S, TargetTerm ([S])) ∈ (indRelRST SRel TRel)*
    using indRelRST.encR[of S SRel TRel]
    by simp
next
  case (source S1 S2)
  assume (S1, S2) ∈ SRel
  thus (SourceTerm S1, SourceTerm S2) ∈ (indRelRST SRel TRel)*
    using indRelRST.source[of S1 S2 SRel TRel]
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  thus (TargetTerm T1, TargetTerm T2) ∈ (indRelRST SRel TRel)*
    using indRelRST.target[of T1 T2 TRel SRel]
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ (indRelRST SRel TRel)* and (Q, R) ∈ (indRelRST SRel TRel)*
  thus (P, R) ∈ (indRelRST SRel TRel)*
    by simp
qed
next
  fix P Q
  assume (P, Q) ∈ (indRelRST SRel TRel)*
  thus P ≲[·]R<SRel,TRel> Q
proof induct
  from reflS reflT show P ≲[·]R<SRel,TRel> P
    using indRelRSTPO-refl[of SRel TRel]
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume P ≲[·]R<SRel,TRel> Q
  moreover assume Q ≲[·]R<SRel,TRel> R
  hence Q ≲[·]R<SRel,TRel> R
    by (induct, simp-all add: indRelRSTPO.intros)
  ultimately show P ≲[·]R<SRel,TRel> R
    by (rule indRelRSTPO.trans)
qed
qed

inductive-set (in encoding) indRelLST
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set
  ⇒ (((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
  for SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  where
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelLST SRel TRel |
  source: (S1, S2) ∈ SRel ⇒ (SourceTerm S1, SourceTerm S2) ∈ indRelLST SRel TRel |
  target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelLST SRel TRel

```

**abbreviation** (in *encoding*) *indRelLSTinfix*

$:: ('procS, 'procT) Proc \Rightarrow ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set$   
 $\Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \mathcal{R}[\cdot]L<-,-> - [75, 75, 75, 75] 80)$

**where**

$P \mathcal{R}[\cdot]L<SRel, TRel> Q \equiv (P, Q) \in indRelLST\ SRel\ TRel$

**inductive-set** (in *encoding*) *indRelLSTPO*

$:: ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set$   
 $\Rightarrow (((('procS, 'procT) Proc) \times ((('procS, 'procT) Proc))) set$

**for** *SRel*  $:: ('procS \times 'procS) set$

**and** *TRel*  $:: ('procT \times 'procT) set$

**where**

*encL*:  $(TargetTerm\ ([S]), SourceTerm\ S) \in indRelLSTPO\ SRel\ TRel \mid$

*source*:  $(S1, S2) \in SRel \Longrightarrow (SourceTerm\ S1, SourceTerm\ S2) \in indRelLSTPO\ SRel\ TRel \mid$

*target*:  $(T1, T2) \in TRel \Longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in indRelLSTPO\ SRel\ TRel \mid$

*trans*:  $[(P, Q) \in indRelLSTPO\ SRel\ TRel; (Q, R) \in indRelLSTPO\ SRel\ TRel]$   
 $\Longrightarrow (P, R) \in indRelLSTPO\ SRel\ TRel$

**abbreviation** (in *encoding*) *indRelLSTPOinfix*

$:: ('procS, 'procT) Proc \Rightarrow ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set$   
 $\Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \lesssim[\cdot]L<-,-> - [75, 75, 75, 75] 80)$

**where**

$P \lesssim[\cdot]L<SRel, TRel> Q \equiv (P, Q) \in indRelLSTPO\ SRel\ TRel$

**lemma** (in *encoding*) *indRelLSTPO-refl*:

**fixes** *SRel*  $:: ('procS \times 'procS) set$

**and** *TRel*  $:: ('procT \times 'procT) set$

**assumes** *reflS*: *refl SRel*

**and** *reflT*: *refl TRel*

**shows** *refl* (*indRelLSTPO SRel TRel*)

**unfolding** *refl-on-def*

**proof** *auto*

**fix** *P*

**show**  $P \lesssim[\cdot]L<SRel, TRel> P$

**proof** (*cases P*)

**case** (*SourceTerm SP*)

**assume**  $SP \in S\ P$

**with** *reflS* **show**  $P \lesssim[\cdot]L<SRel, TRel> P$

**unfolding** *refl-on-def*

**by** (*simp add: indRelLSTPO.source*)

**next**

**case** (*TargetTerm TP*)

**assume**  $TP \in T\ P$

**with** *reflT* **show**  $P \lesssim[\cdot]L<SRel, TRel> P$

**unfolding** *refl-on-def*

**by** (*simp add: indRelLSTPO.target*)

**qed**

**qed**

**lemma** (in *encoding*) *indRelLSTPO-trans*:

**fixes** *SRel*  $:: ('procS \times 'procS) set$

**and** *TRel*  $:: ('procT \times 'procT) set$

**shows** *trans* (*indRelLSTPO SRel TRel*)

**unfolding** *trans-def*

**proof** *clarify*

**fix** *P Q R*

**assume**  $P \lesssim[\cdot]L<SRel, TRel> Q$  **and**  $Q \lesssim[\cdot]L<SRel, TRel> R$

**thus**  $P \lesssim[\cdot]L<SRel, TRel> R$

**by** (*rule indRelLSTPO.trans*)

**qed**

```

lemma (in encoding) refl-trans-closure-of-indRelLST:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes reflS: refl SRel
    and reflT: refl TRel
  shows indRelLSTPO SRel TRel = (indRelLST SRel TRel)*
proof auto
  fix P Q
  assume  $P \lesssim [\cdot]L \langle SRel, TRel \rangle Q$ 
  thus  $(P, Q) \in (indRelLST\ SRel\ TRel)^*$ 
  proof induct
    case (encL S)
    show  $(TargetTerm\ ([S]), SourceTerm\ S) \in (indRelLST\ SRel\ TRel)^*$ 
      using indRelLST.encL[of S SRel TRel]
      by simp
    next
      case (source S1 S2)
      assume  $(S1, S2) \in SRel$ 
      thus  $(SourceTerm\ S1, SourceTerm\ S2) \in (indRelLST\ SRel\ TRel)^*$ 
        using indRelLST.source[of S1 S2 SRel TRel]
        by simp
      next
        case (target T1 T2)
        assume  $(T1, T2) \in TRel$ 
        thus  $(TargetTerm\ T1, TargetTerm\ T2) \in (indRelLST\ SRel\ TRel)^*$ 
          using indRelLST.target[of T1 T2 TRel SRel]
          by simp
        next
          case (trans P Q R)
          assume  $(P, Q) \in (indRelLST\ SRel\ TRel)^*$  and  $(Q, R) \in (indRelLST\ SRel\ TRel)^*$ 
          thus  $(P, R) \in (indRelLST\ SRel\ TRel)^*$ 
            by simp
          qed
        next
          fix P Q
          assume  $(P, Q) \in (indRelLST\ SRel\ TRel)^*$ 
          thus  $P \lesssim [\cdot]L \langle SRel, TRel \rangle Q$ 
          proof induct
            from reflS reflT show  $P \lesssim [\cdot]L \langle SRel, TRel \rangle P$ 
              using indRelLSTPO-refl[of SRel TRel]
              unfolding refl-on-def
              by simp
            next
              case (step Q R)
              assume  $P \lesssim [\cdot]L \langle SRel, TRel \rangle Q$ 
              moreover assume  $Q \mathcal{R} [\cdot]L \langle SRel, TRel \rangle R$ 
              hence  $Q \lesssim [\cdot]L \langle SRel, TRel \rangle R$ 
                by (induct, simp-all add: indRelLSTPO.intros)
              ultimately show  $P \lesssim [\cdot]L \langle SRel, TRel \rangle R$ 
                by (rule indRelLSTPO.trans)
              qed
            qed
          qed

```

```

inductive-set (in encoding) indRelST
  :: ('procS × 'procS) set  $\Rightarrow$  ('procT × 'procT) set
     $\Rightarrow$  (((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
  for SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  where
    encR:  $(SourceTerm\ S, TargetTerm\ ([S])) \in indRelST\ SRel\ TRel$  |
    encL:  $(TargetTerm\ ([S]), SourceTerm\ S) \in indRelST\ SRel\ TRel$  |

```

source:  $(S1, S2) \in SRel \implies (SourceTerm\ S1, SourceTerm\ S2) \in indRelST\ SRel\ TRel \mid$   
target:  $(T1, T2) \in TRel \implies (TargetTerm\ T1, TargetTerm\ T2) \in indRelST\ SRel\ TRel$

**abbreviation** (in encoding) *indRelSTinfix*

$:: ('procS, 'procT)\ Proc \Rightarrow ('procS \times 'procS)\ set \Rightarrow ('procT \times 'procT)\ set$   
 $\Rightarrow ('procS, 'procT)\ Proc \Rightarrow bool\ (-\ \mathcal{R}[\cdot]\langle -, \rangle - [75, 75, 75, 75]\ 80)$

**where**

$P\ \mathcal{R}[\cdot]\langle SRel, TRel \rangle\ Q \equiv (P, Q) \in indRelST\ SRel\ TRel$

**lemma** (in encoding) *indRelST-symm*:

**fixes**  $SRel :: ('procS \times 'procS)\ set$

**and**  $TRel :: ('procT \times 'procT)\ set$

**assumes**  $symmS: sym\ SRel$

**and**  $symmT: sym\ TRel$

**shows**  $sym\ (indRelST\ SRel\ TRel)$

**unfolding** *sym-def*

**proof** *clarify*

**fix**  $P\ Q$

**assume**  $(P, Q) \in indRelST\ SRel\ TRel$

**thus**  $(Q, P) \in indRelST\ SRel\ TRel$

**proof** *induct*

**case**  $(encR\ S)$

**show**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in indRelST\ SRel\ TRel$

**by**  $(rule\ indRelST.encL)$

**next**

**case**  $(encL\ S)$

**show**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelST\ SRel\ TRel$

**by**  $(rule\ indRelST.encR)$

**next**

**case**  $(source\ S1\ S2)$

**assume**  $(S1, S2) \in SRel$

**with**  $symmS$  **show**  $(SourceTerm\ S2, SourceTerm\ S1) \in indRelST\ SRel\ TRel$

**unfolding** *sym-def*

**by**  $(simp\ add: indRelST.source)$

**next**

**case**  $(target\ T1\ T2)$

**assume**  $(T1, T2) \in TRel$

**with**  $symmT$  **show**  $(TargetTerm\ T2, TargetTerm\ T1) \in indRelST\ SRel\ TRel$

**unfolding** *sym-def*

**by**  $(simp\ add: indRelST.target)$

**qed**

**qed**

**inductive-set** (in encoding) *indRelSTEQ*

$:: ('procS \times 'procS)\ set \Rightarrow ('procT \times 'procT)\ set$

$\Rightarrow (((('procS, 'procT)\ Proc) \times (('procS, 'procT)\ Proc))\ set$

**for**  $SRel :: ('procS \times 'procS)\ set$

**and**  $TRel :: ('procT \times 'procT)\ set$

**where**

$encR: (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelSTEQ\ SRel\ TRel \mid$

$encL: (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in indRelSTEQ\ SRel\ TRel \mid$

$source: (S1, S2) \in SRel \implies (SourceTerm\ S1, SourceTerm\ S2) \in indRelSTEQ\ SRel\ TRel \mid$

$target: (T1, T2) \in TRel \implies (TargetTerm\ T1, TargetTerm\ T2) \in indRelSTEQ\ SRel\ TRel \mid$

$trans: \llbracket (P, Q) \in indRelSTEQ\ SRel\ TRel; (Q, R) \in indRelSTEQ\ SRel\ TRel \rrbracket$

$\implies (P, R) \in indRelSTEQ\ SRel\ TRel$

**abbreviation** (in encoding) *indRelSTEQinfix*

$:: ('procS, 'procT)\ Proc \Rightarrow ('procS \times 'procS)\ set \Rightarrow ('procT \times 'procT)\ set$

$\Rightarrow ('procS, 'procT)\ Proc \Rightarrow bool\ (-\ \sim[\cdot]\langle -, \rangle - [75, 75, 75, 75]\ 80)$

**where**

$P\ \sim[\cdot]\langle SRel, TRel \rangle\ Q \equiv (P, Q) \in indRelSTEQ\ SRel\ TRel$

```

lemma (in encoding) indRelSTEQ-refl:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes reflT: refl TRel
  shows refl (indRelSTEQ SRel TRel)
    unfolding refl-on-def
proof auto
  fix P
  show  $P \sim [\cdot] \langle SRel, TRel \rangle P$ 
  proof (cases P)
    case (SourceTerm SP)
      assume  $SP \in S$ 
      moreover have SourceTerm SP  $\sim [\cdot] \langle SRel, TRel \rangle$  TargetTerm ([SP])
        by (rule indRelSTEQ.encR)
      moreover have TargetTerm ([SP])  $\sim [\cdot] \langle SRel, TRel \rangle$  SourceTerm SP
        by (rule indRelSTEQ.encL)
      ultimately show  $P \sim [\cdot] \langle SRel, TRel \rangle P$ 
        by (simp add: indRelSTEQ.trans[where P=SourceTerm SP and Q=TargetTerm ([SP])])
    next
      case (TargetTerm TP)
        assume  $TP \in T$ 
        with reflT show  $P \sim [\cdot] \langle SRel, TRel \rangle P$ 
          unfolding refl-on-def
          by (simp add: indRelSTEQ.target)
  qed
qed

```

```

lemma (in encoding) indRelSTEQ-symm:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes symmS: sym SRel
    and symmT: sym TRel
  shows sym (indRelSTEQ SRel TRel)
    unfolding sym-def
proof clarify
  fix P Q
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
  thus  $Q \sim [\cdot] \langle SRel, TRel \rangle P$ 
  proof induct
    case (encR S)
      show TargetTerm ([S])  $\sim [\cdot] \langle SRel, TRel \rangle$  SourceTerm S
        by (rule indRelSTEQ.encL)
    next
      case (encL S)
        show SourceTerm S  $\sim [\cdot] \langle SRel, TRel \rangle$  TargetTerm ([S])
          by (rule indRelSTEQ.encR)
    next
      case (source S1 S2)
        assume  $(S1, S2) \in SRel$ 
        with symmS show SourceTerm S2  $\sim [\cdot] \langle SRel, TRel \rangle$  SourceTerm S1
          unfolding sym-def
          by (simp add: indRelSTEQ.source)
    next
      case (target T1 T2)
        assume  $(T1, T2) \in TRel$ 
        with symmT show TargetTerm T2  $\sim [\cdot] \langle SRel, TRel \rangle$  TargetTerm T1
          unfolding sym-def
          by (simp add: indRelSTEQ.target)
    next
      case (trans P Q R)

```

```

    assume  $R \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle P$ 
    thus  $R \sim [\cdot] \langle SRel, TRel \rangle P$ 
    by (rule indRelSTEQ.trans)
qed
qed

lemma (in encoding) indRelSTEQ-trans:
  fixes  $SRel :: ('procS \times 'procS)$  set
    and  $TRel :: ('procT \times 'procT)$  set
  shows trans (indRelSTEQ SRel TRel)
    unfolding trans-def
proof clarify
  fix  $P Q R$ 
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
    by (rule indRelSTEQ.trans)
qed

lemma (in encoding) refl-trans-closure-of-indRelST:
  fixes  $SRel :: ('procS \times 'procS)$  set
    and  $TRel :: ('procT \times 'procT)$  set
  assumes reflT: refl TRel
  shows indRelSTEQ SRel TRel = (indRelST SRel TRel)*
proof auto
  fix  $P Q$ 
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
  thus  $(P, Q) \in (indRelST SRel TRel)^*$ 
proof induct
  case (encR S)
  show (SourceTerm S, TargetTerm ([S]))  $\in (indRelST SRel TRel)^*$ 
    using indRelST.encR[of S SRel TRel]
    by simp
next
  case (encL S)
  show (TargetTerm ([S]), SourceTerm S)  $\in (indRelST SRel TRel)^*$ 
    using indRelST.encL[of S SRel TRel]
    by simp
next
  case (source S1 S2)
  assume  $(S1, S2) \in SRel$ 
  thus (SourceTerm S1, SourceTerm S2)  $\in (indRelST SRel TRel)^*$ 
    using indRelST.source[of S1 S2 SRel TRel]
    by simp
next
  case (target T1 T2)
  assume  $(T1, T2) \in TRel$ 
  thus (TargetTerm T1, TargetTerm T2)  $\in (indRelST SRel TRel)^*$ 
    using indRelST.target[of T1 T2 TRel SRel]
    by simp
next
  case (trans P Q R)
  assume  $(P, Q) \in (indRelST SRel TRel)^*$  and  $(Q, R) \in (indRelST SRel TRel)^*$ 
  thus  $(P, R) \in (indRelST SRel TRel)^*$ 
    by simp
qed
next
  fix  $P Q$ 
  assume  $(P, Q) \in (indRelST SRel TRel)^*$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
proof induct
  from reflT show  $P \sim [\cdot] \langle SRel, TRel \rangle P$ 

```

```

    using indRelSTEQ-refl[of TRel SRel]
    unfolding refl-on-def
  by simp
next
case (step Q R)
assume P ~[·]<SRel,TRel> Q
moreover assume Q  $\mathcal{R}$ [·]<SRel,TRel> R
hence Q ~[·]<SRel,TRel> R
  by (induct, simp-all add: indRelSTEQ.intros)
ultimately show P ~[·]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
qed
qed

lemma (in encoding) refl-symm-trans-closure-of-indRelST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
and symmS: sym SRel
and symmT: sym TRel
shows indRelSTEQ SRel TRel = (symcl ((indRelST SRel TRel)=))+
proof -
have (symcl ((indRelST SRel TRel)=))+ = (symcl (indRelST SRel TRel))*
  by (rule refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelST SRel TRel])
moreover from symmS symmT have symcl (indRelST SRel TRel) = indRelST SRel TRel
  using indRelST-symm[where SRel=SRel and TRel=TRel]
  symm-closure-of-symm-rel[where Rel=indRelST SRel TRel]
  by blast
ultimately show indRelSTEQ SRel TRel = (symcl ((indRelST SRel TRel)=))+
  using reflT refl-trans-closure-of-indRelST[where SRel=SRel and TRel=TRel]
  by simp
qed

lemma (in encoding) symm-closure-of-indRelRST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
and symmS: sym SRel
and symmT: sym TRel
shows indRelRST SRel TRel = symcl (indRelRST SRel TRel)
and indRelSTEQ SRel TRel = (symcl ((indRelRST SRel TRel)=))+
proof -
show indRelRST SRel TRel = symcl (indRelRST SRel TRel)
proof auto
fix P Q
assume P  $\mathcal{R}$ [·]<SRel,TRel> Q
thus (P, Q) ∈ symcl (indRelRST SRel TRel)
  by (induct, simp-all add: symcl-def indRelRST.intros)
next
fix P Q
assume (P, Q) ∈ symcl (indRelRST SRel TRel)
thus P  $\mathcal{R}$ [·]<SRel,TRel> Q
proof (auto simp add: symcl-def indRelRST.simps)
fix S
show SourceTerm S  $\mathcal{R}$ [·]<SRel,TRel> TargetTerm ([S])
  by (rule indRelST.encR)
next
fix S1 S2
assume (S1, S2) ∈ SRel
thus SourceTerm S1  $\mathcal{R}$ [·]<SRel,TRel> SourceTerm S2
  by (rule indRelST.source)

```



```

next
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1  $\mathcal{R}[\cdot]$ <SRel,TRel> TargetTerm T2
    by (rule indRelST.target)
next
  fix S
  show TargetTerm ( $\llbracket S \rrbracket$ )  $\mathcal{R}[\cdot]$ <SRel,TRel> SourceTerm S
    by (rule indRelST.encL)
next
  fix S1 S2
  assume (S1, S2) ∈ SRel
  with symmS show SourceTerm S2  $\mathcal{R}[\cdot]$ <SRel,TRel> SourceTerm S1
    unfolding sym-def
    by (simp add: indRelST.source)
next
  fix T1 T2
  assume (T1, T2) ∈ TRel
  with symmT show (TargetTerm T2, TargetTerm T1) ∈ indRelST SRel TRel
    unfolding sym-def
    by (simp add: indRelST.target)
qed
qed
with reflT show indRelSTEQ SRel TRel = (symcl ((indRelRST SRel TRel)=))+
  using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelRST SRel TRel]
  refl-trans-closure-of-indRelST
  by simp
qed

lemma (in encoding) symm-closure-of-indRelLST:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes reflT: refl TRel
  and symmS: sym SRel
  and symmT: sym TRel
  shows indRelST SRel TRel = symcl (indRelLST SRel TRel)
  and indRelSTEQ SRel TRel = (symcl ((indRelLST SRel TRel)=))+
proof -
  show indRelST SRel TRel = symcl (indRelLST SRel TRel)
proof auto
  fix P Q
  assume P  $\mathcal{R}[\cdot]$ <SRel,TRel> Q
  thus (P, Q) ∈ symcl (indRelLST SRel TRel)
    by (induct, simp-all add: symcl-def indRelLST.intros)
next
  fix P Q
  assume (P, Q) ∈ symcl (indRelLST SRel TRel)
  thus P  $\mathcal{R}[\cdot]$ <SRel,TRel> Q
proof (auto simp add: symcl-def indRelLST.simps)
  fix S
  show SourceTerm S  $\mathcal{R}[\cdot]$ <SRel,TRel> TargetTerm ( $\llbracket S \rrbracket$ )
    by (rule indRelST.encR)
next
  fix S1 S2
  assume (S1, S2) ∈ SRel
  thus SourceTerm S1  $\mathcal{R}[\cdot]$ <SRel,TRel> SourceTerm S2
    by (rule indRelST.source)
next
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1  $\mathcal{R}[\cdot]$ <SRel,TRel> TargetTerm T2

```

```

    by (rule indRelST.target)
next
fix S
show TargetTerm ([[S]])  $\mathcal{R}[\cdot]$ <SRel,TRel> SourceTerm S
  by (rule indRelST.encL)
next
fix S1 S2
assume (S1, S2)  $\in$  SRel
with symmS show SourceTerm S2  $\mathcal{R}[\cdot]$ <SRel,TRel> SourceTerm S1
  unfolding sym-def
  by (simp add: indRelST.source)
next
fix T1 T2
assume (T1, T2)  $\in$  TRel
with symmT show TargetTerm T2  $\mathcal{R}[\cdot]$ <SRel,TRel> TargetTerm T1
  unfolding sym-def
  by (simp add: indRelST.target)
qed
qed
with reflT show indRelSTEQ SRel TRel = (symcl ((indRelLST SRel TRel)=))+
  using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelLST SRel TRel]
  refl-trans-closure-of-indRelST
  by simp
qed

```

lemma (in encoding) symm-trans-closure-of-indRelRSTPO:

```

fixes SRel :: ('procS  $\times$  'procS) set
and TRel :: ('procT  $\times$  'procT) set
assumes symmS: sym SRel
and symmT: sym TRel
shows indRelSTEQ SRel TRel = (symcl (indRelRSTPO SRel TRel))+
proof auto
fix P Q
assume P  $\sim$ [[ $\cdot$ ]]<SRel,TRel> Q
thus (P, Q)  $\in$  (symcl (indRelRSTPO SRel TRel))+
proof induct
case (encR S)
show (SourceTerm S, TargetTerm ([[S]]))  $\in$  (symcl (indRelRSTPO SRel TRel))+
  using indRelRSTPO.encR[of S SRel TRel]
  unfolding symcl-def
  by auto
next
case (encL S)
show (TargetTerm ([[S]]), SourceTerm S)  $\in$  (symcl (indRelRSTPO SRel TRel))+
  using indRelRSTPO.encR[of S SRel TRel]
  unfolding symcl-def
  by auto
next
case (source S1 S2)
assume (S1, S2)  $\in$  SRel
thus (SourceTerm S1, SourceTerm S2)  $\in$  (symcl (indRelRSTPO SRel TRel))+
  using indRelRSTPO.source[of S1 S2 SRel TRel]
  unfolding symcl-def
  by auto
next
case (target T1 T2)
assume (T1, T2)  $\in$  TRel
thus (TargetTerm T1, TargetTerm T2)  $\in$  (symcl (indRelRSTPO SRel TRel))+
  using indRelRSTPO.target[of T1 T2 TRel SRel]
  unfolding symcl-def
  by auto

```

```

next
  case (trans P Q R)
  assume (P, Q) ∈ (symcl (indRelRSTPO SRel TRel))+
  and (Q, R) ∈ (symcl (indRelRSTPO SRel TRel))+
  thus (P, R) ∈ (symcl (indRelRSTPO SRel TRel))+
  by simp
qed
next
fix P Q
assume (P, Q) ∈ (symcl (indRelRSTPO SRel TRel))+
thus P ∼[·]<SRel, TRel> Q
proof induct
  fix Q
  assume (P, Q) ∈ symcl (indRelRSTPO SRel TRel)
  thus P ∼[·]<SRel, TRel> Q
  proof (cases P ∼[·]R<SRel, TRel> Q, simp-all add: symcl-def)
    assume P ∼[·]R<SRel, TRel> Q
    thus P ∼[·]<SRel, TRel> Q
    proof induct
      case (encR S)
      show SourceTerm S ∼[·]<SRel, TRel> TargetTerm ([S])
      by (rule indRelSTEQ.encR)
    next
      case (source S1 S2)
      assume (S1, S2) ∈ SRel
      thus SourceTerm S1 ∼[·]<SRel, TRel> SourceTerm S2
      by (rule indRelSTEQ.source)
    next
      case (target T1 T2)
      assume (T1, T2) ∈ TRel
      thus TargetTerm T1 ∼[·]<SRel, TRel> TargetTerm T2
      by (rule indRelSTEQ.target)
    next
      case (trans P Q R)
      assume P ∼[·]<SRel, TRel> Q and Q ∼[·]<SRel, TRel> R
      thus P ∼[·]<SRel, TRel> R
      by (rule indRelSTEQ.trans)
  qed
next
assume Q ∼[·]R<SRel, TRel> P
thus P ∼[·]<SRel, TRel> Q
proof induct
  case (encR S)
  show TargetTerm ([S]) ∼[·]<SRel, TRel> SourceTerm S
  by (rule indRelSTEQ.encL)
next
  case (source S1 S2)
  assume (S1, S2) ∈ SRel
  with symmS show SourceTerm S2 ∼[·]<SRel, TRel> SourceTerm S1
  unfolding sym-def
  by (simp add: indRelSTEQ.source)
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  with symmT show TargetTerm T2 ∼[·]<SRel, TRel> TargetTerm T1
  unfolding sym-def
  by (simp add: indRelSTEQ.target)
next
  case (trans P Q R)
  assume R ∼[·]<SRel, TRel> Q and Q ∼[·]<SRel, TRel> P
  thus R ∼[·]<SRel, TRel> P

```

```

    by (rule indRelSTEQ.trans)
  qed
qed
next
case (step Q R)
assume P ~[[·]]<SRel,TRel> Q
moreover assume (Q, R) ∈ symcl (indRelRSTPO SRel TRel)
hence Q ~[[·]]<SRel,TRel> R
proof (auto simp add: symcl-def)
  assume Q ≲[[·]]R<SRel,TRel> R
  thus Q ~[[·]]<SRel,TRel> R
  proof (induct, simp add: indRelSTEQ.encR, simp add: indRelSTEQ.source,
    simp add: indRelSTEQ.target)
    case (trans P Q R)
    assume P ~[[·]]<SRel,TRel> Q and Q ~[[·]]<SRel,TRel> R
    thus P ~[[·]]<SRel,TRel> R
    by (rule indRelSTEQ.trans)
  qed
qed
next
assume R ≲[[·]]R<SRel,TRel> Q
hence R ~[[·]]<SRel,TRel> Q
proof (induct, simp add: indRelSTEQ.encR, simp add: indRelSTEQ.source,
  simp add: indRelSTEQ.target)
  case (trans P Q R)
  assume P ~[[·]]<SRel,TRel> Q and Q ~[[·]]<SRel,TRel> R
  thus P ~[[·]]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
qed
with symmS symmT show Q ~[[·]]<SRel,TRel> R
  using indRelSTEQ-symm[of SRel TRel]
  unfolding sym-def
  by blast
qed
ultimately show P ~[[·]]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
qed
qed

lemma (in encoding) symm-trans-closure-of-indRelLSTPO:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes symmS: sym SRel
  and symmT: sym TRel
  shows indRelSTEQ SRel TRel = (symcl (indRelLSTPO SRel TRel))+
proof auto
  fix P Q
  assume P ~[[·]]<SRel,TRel> Q
  thus (P, Q) ∈ (symcl (indRelLSTPO SRel TRel))+
  proof induct
    case (encR S)
    show (SourceTerm S, TargetTerm ([[S]]) ∈ (symcl (indRelLSTPO SRel TRel))+
      using indRelLSTPO.encL[of S SRel TRel]
      unfolding symcl-def
      by blast
    next
    case (encL S)
    show (TargetTerm ([[S]]), SourceTerm S) ∈ (symcl (indRelLSTPO SRel TRel))+
      using indRelLSTPO.encL[of S SRel TRel]
      unfolding symcl-def
      by blast
  next
  next

```

```

case (source  $S1\ S2$ )
assume ( $S1, S2$ )  $\in SRel$ 
thus ( $SourceTerm\ S1, SourceTerm\ S2$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
  using  $indRelLSTPO.source[of\ S1\ S2\ SRel\ TRel]$ 
  unfolding  $symcl-def$ 
  by  $blast$ 
next
case (target  $T1\ T2$ )
assume ( $T1, T2$ )  $\in TRel$ 
thus ( $TargetTerm\ T1, TargetTerm\ T2$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
  using  $indRelLSTPO.target[of\ T1\ T2\ TRel\ SRel]$ 
  unfolding  $symcl-def$ 
  by  $blast$ 
next
case (trans  $P\ Q\ R$ )
assume ( $P, Q$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
  and ( $Q, R$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
thus ( $P, R$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
  by  $simp$ 
qed
next
fix  $P\ Q$ 
assume ( $P, Q$ )  $\in (symcl\ (indRelLSTPO\ SRel\ TRel))^+$ 
thus  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
proof  $induct$ 
  fix  $Q$ 
assume ( $P, Q$ )  $\in symcl\ (indRelLSTPO\ SRel\ TRel)$ 
thus  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
  unfolding  $symcl-def$ 
proof  $auto$ 
  assume  $P \lesssim [\cdot] L \langle SRel, TRel \rangle Q$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
  proof ( $induct, simp\ add: indRelSTEQ.encL, simp\ add: indRelSTEQ.source,$ 
     $simp\ add: indRelSTEQ.target$ )
    case (trans  $P\ Q\ R$ )
    assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
    thus  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
    by ( $rule\ indRelSTEQ.trans$ )
  qed
next
assume  $Q \lesssim [\cdot] L \langle SRel, TRel \rangle P$ 
hence  $Q \sim [\cdot] \langle SRel, TRel \rangle P$ 
proof ( $induct, simp\ add: indRelSTEQ.encL, simp\ add: indRelSTEQ.source,$ 
   $simp\ add: indRelSTEQ.target$ )
  case (trans  $P\ Q\ R$ )
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
  by ( $rule\ indRelSTEQ.trans$ )
qed
with  $symmS\ symmT$  show  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
  using  $indRelSTEQ-symm[of\ SRel\ TRel]$ 
  unfolding  $sym-def$ 
  by  $blast$ 
qed
next
case (step  $Q\ R$ )
assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$ 
moreover assume ( $Q, R$ )  $\in symcl\ (indRelLSTPO\ SRel\ TRel)$ 
hence  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  unfolding  $symcl-def$ 
proof  $auto$ 

```

```

assume  $Q \lesssim [\cdot] L \langle SRel, TRel \rangle R$ 
thus  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ.source,
        simp add: indRelSTEQ.target)
  case (trans P Q R)
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
  by (rule indRelSTEQ.trans)
qed
next
assume  $R \lesssim [\cdot] L \langle SRel, TRel \rangle Q$ 
hence  $R \sim [\cdot] \langle SRel, TRel \rangle Q$ 
proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ.source,
        simp add: indRelSTEQ.target)
  case (trans P Q R)
  assume  $P \sim [\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  thus  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
  by (rule indRelSTEQ.trans)
qed
with symmS symmT show  $Q \sim [\cdot] \langle SRel, TRel \rangle R$ 
  using indRelSTEQ-symm[of SRel TRel]
  unfolding sym-def
  by blast
qed
ultimately show  $P \sim [\cdot] \langle SRel, TRel \rangle R$ 
by (rule indRelSTEQ.trans)
qed
qed

```

If the relations `indRelRST`, `indRelLST`, or `indRelST` contain a pair of target terms, then this pair is also related by the considered target term relation. Similarly a pair of source terms is related by the considered source term relation.

```

lemma (in encoding) indRelRST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP  $\mathcal{R}[\cdot] R \langle SRel, TRel \rangle$  SourceTerm SQ
  shows (SP, SQ) ∈ SRel
  using rel
  by (simp add: indRelRST.simps)

```

```

lemma (in encoding) indRelRST-to-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  assumes rel: TargetTerm TP  $\mathcal{R}[\cdot] R \langle SRel, TRel \rangle$  TargetTerm TQ
  shows (TP, TQ) ∈ TRel
  using rel
  by (simp add: indRelRST.simps)

```

```

lemma (in encoding) indRelLST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP  $\mathcal{R}[\cdot] L \langle SRel, TRel \rangle$  SourceTerm SQ
  shows (SP, SQ) ∈ SRel
  using rel
  by (simp add: indRelLST.simps)

```

```

lemma (in encoding) indRelLST-to-TRel:

```

```

fixes  $SRel$  :: ('procS × 'procS) set
and  $TRel$  :: ('procT × 'procT) set
and  $TP\ TQ$  :: 'procT
assumes  $rel$ : TargetTerm  $TP\ \mathcal{R}[\cdot]$  L< $SRel, TRel$ > TargetTerm  $TQ$ 
shows  $(TP, TQ) \in TRel$ 
using  $rel$ 
by (simp add: indRelLST.simps)

```

```

lemma (in encoding) indRelST-to-SRel:
fixes  $SRel$  :: ('procS × 'procS) set
and  $TRel$  :: ('procT × 'procT) set
and  $SP\ SQ$  :: 'procS
assumes  $rel$ : SourceTerm  $SP\ \mathcal{R}[\cdot]$  < $SRel, TRel$ > SourceTerm  $SQ$ 
shows  $(SP, SQ) \in SRel$ 
using  $rel$ 
by (simp add: indRelST.simps)

```

```

lemma (in encoding) indRelST-to-TRel:
fixes  $SRel$  :: ('procS × 'procS) set
and  $TRel$  :: ('procT × 'procT) set
and  $TP\ TQ$  :: 'procT
assumes  $rel$ : TargetTerm  $TP\ \mathcal{R}[\cdot]$  < $SRel, TRel$ > TargetTerm  $TQ$ 
shows  $(TP, TQ) \in TRel$ 
using  $rel$ 
by (simp add: indRelST.simps)

```

If the relations  $indRelRSTPO$  or  $indRelLSTPO$  contain a pair of target terms, then this pair is also related by the transitive closure of the considered target term relation. Similarly a pair of source terms is related by the transitive closure of the source term relation. A pair of a source and a target term results from the combination of pairs in the source relation, the target relation, and the encoding function. Note that, because of the symmetry, no similar condition holds for  $indRelSTEQ$ .

```

lemma (in encoding) indRelRSTPO-to-SRel-and-TRel:
fixes  $SRel$  :: ('procS × 'procS) set
and  $TRel$  :: ('procT × 'procT) set
and  $P\ Q$  :: ('procS, 'procT) Proc
assumes  $P \lesssim[\cdot] R$  < $SRel, TRel$ >  $Q$ 
shows  $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (SP, SQ) \in SRel^+$ 
and  $\forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\exists S. (SP, S) \in SRel^* \wedge ([S], TQ) \in TRel^*)$ 
and  $\forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow False$ 
and  $\forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel^+$ 
using  $assms$ 
proof induct
case (encR  $S$ )
show  $\forall SP\ SQ. SP \in S\ SourceTerm\ S \wedge SQ \in S\ TargetTerm\ ([S]) \longrightarrow (SP, SQ) \in SRel^+$ 
and  $\forall TP\ TQ. TP \in T\ SourceTerm\ S \wedge TQ \in S\ TargetTerm\ ([S]) \longrightarrow False$ 
and  $\forall TP\ TQ. TP \in T\ SourceTerm\ S \wedge TQ \in T\ TargetTerm\ ([S]) \longrightarrow (TP, TQ) \in TRel^+$ 
by simp+
have  $(S, S) \in SRel^*$ 
by simp
moreover have  $([S], [S]) \in TRel^*$ 
by simp
ultimately show  $\forall SP\ TQ. SP \in S\ SourceTerm\ S \wedge TQ \in T\ TargetTerm\ ([S]) \longrightarrow$ 
 $(\exists S. (SP, S) \in SRel^* \wedge ([S], TQ) \in TRel^*)$ 
by blast
next
case (source  $S1\ S2$ )
assume  $(S1, S2) \in SRel$ 
thus  $\forall SP\ SQ. SP \in S\ SourceTerm\ S1 \wedge SQ \in S\ SourceTerm\ S2 \longrightarrow (SP, SQ) \in SRel^+$ 
by simp
show  $\forall SP\ TQ. SP \in S\ SourceTerm\ S1 \wedge TQ \in T\ SourceTerm\ S2 \longrightarrow$ 

```

$(\exists S. (SP, S) \in SRel^* \wedge ([S], TQ) \in TRel^*)$   
**and**  $\forall TP SQ. TP \in T \text{ SourceTerm } S1 \wedge SQ \in S \text{ SourceTerm } S2 \longrightarrow \text{False}$   
**and**  $\forall TP TQ. TP \in T \text{ SourceTerm } S1 \wedge TQ \in T \text{ SourceTerm } S2 \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp+*  
**next**  
**case** (*target*  $T1 T2$ )  
**show**  $\forall SP SQ. SP \in S \text{ TargetTerm } T1 \wedge SQ \in S \text{ TargetTerm } T2 \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $\forall SP TQ. SP \in S \text{ TargetTerm } T1 \wedge TQ \in T \text{ TargetTerm } T2$   
 $\longrightarrow (\exists S. (SP, S) \in SRel^* \wedge ([S], TQ) \in TRel^*)$   
**and**  $\forall TP SQ. TP \in T \text{ TargetTerm } T1 \wedge SQ \in S \text{ TargetTerm } T2 \longrightarrow \text{False}$   
**by** *simp+*  
**assume**  $(T1, T2) \in TRel$   
**thus**  $\forall TP TQ. TP \in T \text{ TargetTerm } T1 \wedge TQ \in T \text{ TargetTerm } T2 \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp*  
**next**  
**case** (*trans*  $P Q R$ )  
**assume**  $A1: \forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $A2: \forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\exists S. (SP, S) \in SRel^* \wedge ([S], TQ) \in TRel^*)$   
**and**  $A3: \forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow \text{False}$   
**and**  $A4: \forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$   
**and**  $A5: \forall SQ SR. SQ \in S Q \wedge SR \in S R \longrightarrow (SQ, SR) \in SRel^+$   
**and**  $A6: \forall SQ TR. SQ \in S Q \wedge TR \in T R \longrightarrow (\exists S. (SQ, S) \in SRel^* \wedge ([S], TR) \in TRel^*)$   
**and**  $A7: \forall TQ SR. TQ \in T Q \wedge SR \in S R \longrightarrow \text{False}$   
**and**  $A8: \forall TQ TR. TQ \in T Q \wedge TR \in T R \longrightarrow (TQ, TR) \in TRel^+$   
**show**  $\forall SP SR. SP \in S P \wedge SR \in S R \longrightarrow (SP, SR) \in SRel^+$   
**proof** *clarify*  
**fix**  $SP SR$   
**assume**  $A9: SP \in S P$  **and**  $A10: SR \in S R$   
**show**  $(SP, SR) \in SRel^+$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $A11: SQ \in S Q$   
**with**  $A1 A9$  **have**  $(SP, SQ) \in SRel^+$   
**by** *simp*  
**moreover from**  $A5 A10 A11$  **have**  $(SQ, SR) \in SRel^+$   
**by** *simp*  
**ultimately show**  $(SP, SR) \in SRel^+$   
**by** *simp*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $TQ \in T Q$   
**with**  $A7 A10$  **show**  $(SP, SR) \in SRel^+$   
**by** *blast*  
**qed**  
**qed**  
**show**  $\forall SP TR. SP \in S P \wedge TR \in T R$   
 $\longrightarrow (\exists S. (SP, S) \in SRel^* \wedge ([S], TR) \in TRel^*)$   
**proof** *clarify*  
**fix**  $SP TR$   
**assume**  $A9: SP \in S P$  **and**  $A10: TR \in T R$   
**show**  $\exists S. (SP, S) \in SRel^* \wedge ([S], TR) \in TRel^*$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $A11: SQ \in S Q$   
**with**  $A6 A10$  **obtain**  $S$  **where**  $A12: (SQ, S) \in SRel^*$   
**and**  $A13: ([S], TR) \in TRel^*$   
**by** *blast*  
**from**  $A1 A9 A11$  **have**  $(SP, SQ) \in SRel^*$   
**by** *simp*  
**from** *this*  $A12$  **have**  $(SP, S) \in SRel^*$   
**by** *simp*



```

with A13 show  $\exists S. (SP, S) \in SRel^* \wedge (\llbracket S \rrbracket, TR) \in TRel^*$ 
  by blast
next
case (TargetTerm TQ)
assume A11:  $TQ \in T Q$ 
with A2 A9 obtain S where A12:  $(SP, S) \in SRel^*$ 
  and A13:  $(\llbracket S \rrbracket, TQ) \in TRel^*$ 
  by blast
from A8 A10 A11 have  $(TQ, TR) \in TRel^*$ 
  by simp
with A13 have  $(\llbracket S \rrbracket, TR) \in TRel^*$ 
  by simp
with A12 show  $\exists S. (SP, S) \in SRel^* \wedge (\llbracket S \rrbracket, TR) \in TRel^*$ 
  by blast
qed
qed
show  $\forall TP SR. TP \in T P \wedge SR \in S R \longrightarrow False$ 
proof clarify
  fix TP SR
  assume A9:  $TP \in T P$  and A10:  $SR \in S R$ 
  show False
  proof (cases Q)
    case (SourceTerm SQ)
    assume  $SQ \in S Q$ 
    with A3 A9 show False
      by blast
  next
  case (TargetTerm TQ)
  assume  $TQ \in T Q$ 
  with A7 A10 show False
    by blast
  qed
qed
show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
proof clarify
  fix TP TR
  assume A9:  $TP \in T P$  and A10:  $TR \in T R$ 
  show  $(TP, TR) \in TRel^+$ 
  proof (cases Q)
    case (SourceTerm SQ)
    assume  $SQ \in S Q$ 
    with A3 A9 show  $(TP, TR) \in TRel^+$ 
      by blast
  next
  case (TargetTerm TQ)
  assume A11:  $TQ \in T Q$ 
  with A4 A9 have  $(TP, TQ) \in TRel^+$ 
    by simp
  moreover from A8 A10 A11 have  $(TQ, TR) \in TRel^+$ 
    by simp
  ultimately show  $(TP, TR) \in TRel^+$ 
    by simp
  qed
qed
qed

```

**lemma** (in *encoding*) *indRelLSTPO-to-SRel-and-TRel*:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and P Q :: ('procS, 'procT) Proc
assumes P  $\lesssim$   $\llbracket \cdot \rrbracket L \langle SRel, TRel \rangle Q$ 

```

**shows**  $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow False$   
**and**  $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$   
**and**  $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$   
**using** *assms*  
**proof** *induct*  
**case** (*encL S*)  
**show**  $\forall SP SQ. SP \in S TargetTerm (\llbracket S \rrbracket) \wedge SQ \in S SourceTerm S \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $\forall SP TQ. SP \in S TargetTerm (\llbracket S \rrbracket) \wedge TQ \in T SourceTerm S \longrightarrow False$   
**and**  $\forall TP TQ. TP \in T TargetTerm (\llbracket S \rrbracket) \wedge TQ \in T SourceTerm S \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp+*  
**have**  $(\llbracket S \rrbracket, \llbracket S \rrbracket) \in TRel^*$   
**by** *simp*  
**moreover** **have**  $(S, S) \in SRel^*$   
**by** *simp*  
**ultimately** **show**  $\forall TP SQ. TP \in T TargetTerm (\llbracket S \rrbracket) \wedge SQ \in S SourceTerm S \longrightarrow$   
 $(\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$   
**by** *blast*  
**next**  
**case** (*source S1 S2*)  
**assume**  $(S1, S2) \in SRel$   
**thus**  $\forall SP SQ. SP \in S SourceTerm S1 \wedge SQ \in S SourceTerm S2 \longrightarrow (SP, SQ) \in SRel^+$   
**by** *simp*  
**show**  $\forall SP TQ. SP \in S SourceTerm S1 \wedge TQ \in T SourceTerm S2 \longrightarrow False$   
**and**  $\forall TP SQ. TP \in T SourceTerm S1 \wedge SQ \in S SourceTerm S2$   
 $\longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$   
**and**  $\forall TP TQ. TP \in T SourceTerm S1 \wedge TQ \in T SourceTerm S2 \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp+*  
**next**  
**case** (*target T1 T2*)  
**show**  $\forall SP SQ. SP \in S TargetTerm T1 \wedge SQ \in S TargetTerm T2 \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $\forall SP TQ. SP \in S TargetTerm T1 \wedge TQ \in T TargetTerm T2 \longrightarrow False$   
**and**  $\forall TP SQ. TP \in T TargetTerm T1 \wedge SQ \in S TargetTerm T2$   
 $\longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$   
**by** *simp+*  
**assume**  $(T1, T2) \in TRel$   
**thus**  $\forall TP TQ. TP \in T TargetTerm T1 \wedge TQ \in T TargetTerm T2 \longrightarrow (TP, TQ) \in TRel^+$   
**by** *simp*  
**next**  
**case** (*trans P Q R*)  
**assume**  $A1: \forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+$   
**and**  $A2: \forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow False$   
**and**  $A3: \forall TP SQ. TP \in T P \wedge SQ \in S Q$   
 $\longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$   
**and**  $A4: \forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$   
**and**  $A5: \forall SQ SR. SQ \in S Q \wedge SR \in S R \longrightarrow (SQ, SR) \in SRel^+$   
**and**  $A6: \forall SQ TR. SQ \in S Q \wedge TR \in T R \longrightarrow False$   
**and**  $A7: \forall TQ SR. TQ \in T Q \wedge SR \in S R \longrightarrow (\exists S. (TQ, \llbracket S \rrbracket) \in TRel^* \wedge (S, SR) \in SRel^*)$   
**and**  $A8: \forall TQ TR. TQ \in T Q \wedge TR \in T R \longrightarrow (TQ, TR) \in TRel^+$   
**show**  $\forall SP SR. SP \in S P \wedge SR \in S R \longrightarrow (SP, SR) \in SRel^+$   
**proof** *clarify*  
**fix**  $SP SR$   
**assume**  $A9: SP \in S P$  **and**  $A10: SR \in S R$   
**show**  $(SP, SR) \in SRel^+$   
**proof** (*cases Q*)  
**case** (*SourceTerm SQ*)  
**assume**  $A11: SQ \in S Q$   
**with**  $A1 A9$  **have**  $(SP, SQ) \in SRel^+$   
**by** *simp*  
**moreover** **from**  $A5 A10 A11$  **have**  $(SQ, SR) \in SRel^+$   
**by** *simp*

```

ultimately show  $(SP, SR) \in SRel^+$ 
  by simp
next
case (TargetTerm TQ)
  assume  $TQ \in T Q$ 
  with A2 A9 show  $(SP, SR) \in SRel^+$ 
    by blast
qed
qed
show  $\forall SP TR. SP \in S P \wedge TR \in T R \longrightarrow False$ 
proof clarify
  fix SP TR
  assume A9:  $SP \in S P$  and A10:  $TR \in T R$ 
  show False
proof (cases Q)
  case (SourceTerm SQ)
    assume  $SQ \in S Q$ 
    with A6 A10 show False
      by blast
next
  case (TargetTerm TQ)
    assume  $TQ \in T Q$ 
    with A2 A9 show False
      by blast
qed
qed
show  $\forall TP SR. TP \in T P \wedge SR \in S R \longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SR) \in SRel^*)$ 
proof clarify
  fix TP SR
  assume A9:  $TP \in T P$  and A10:  $SR \in S R$ 
  show  $\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SR) \in SRel^*$ 
proof (cases Q)
  case (SourceTerm SQ)
    assume A11:  $SQ \in S Q$ 
    with A3 A9 obtain S where A12:  $(TP, \llbracket S \rrbracket) \in TRel^*$  and A13:  $(S, SQ) \in SRel^*$ 
      by blast
    from A5 A10 A11 have  $(SQ, SR) \in SRel^*$ 
      by simp
    with A13 have  $(S, SR) \in SRel^*$ 
      by simp
    with A12 show  $\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SR) \in SRel^*$ 
      by blast
next
  case (TargetTerm TQ)
    assume A11:  $TQ \in T Q$ 
    with A7 A10 obtain S where A12:  $(TQ, \llbracket S \rrbracket) \in TRel^*$  and A13:  $(S, SR) \in SRel^*$ 
      by blast
    from A4 A9 A11 have  $(TP, TQ) \in TRel^*$ 
      by simp
    from this A12 have  $(TP, \llbracket S \rrbracket) \in TRel^*$ 
      by simp
    with A13 show  $\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SR) \in SRel^*$ 
      by blast
qed
qed
show  $\forall TP TR. TP \in T P \wedge TR \in T R \longrightarrow (TP, TR) \in TRel^+$ 
proof clarify
  fix TP TR
  assume A9:  $TP \in T P$  and A10:  $TR \in T R$ 
  show  $(TP, TR) \in TRel^+$ 
proof (cases Q)

```

```

  case (SourceTerm SQ)
  assume SQ ∈ S Q
  with A6 A10 show (TP, TR) ∈ TRel+
    by blast
next
  case (TargetTerm TQ)
  assume A11: TQ ∈ T Q
  with A4 A9 have (TP, TQ) ∈ TRel+
    by simp
  moreover from A8 A10 A11 have (TQ, TR) ∈ TRel+
    by simp
  ultimately show (TP, TR) ∈ TRel+
    by simp
qed
qed
qed

```

If  $\text{indRelRSTPO}$ ,  $\text{indRelLSTPO}$ , or  $\text{indRelSTPO}$  preserves barbs then so do the corresponding source term and target term relations.

**lemma** (in *encoding-wrt-barbs*) *rel-with-source-impl-SRel-preserves-barbs*:

```

fixes SRel :: ('procS × 'procS) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
assumes preservation: rel-preserves-barbs Rel (STCalWB SWB TWB)
  and sourceInRel: ∀ S1 S2. (S1, S2) ∈ SRel → (SourceTerm S1, SourceTerm S2) ∈ Rel
shows rel-preserves-barbs SRel SWB

```

**proof** *clarify*

```

fix SP SQ a
assume (SP, SQ) ∈ SRel
with sourceInRel have (SourceTerm SP, SourceTerm SQ) ∈ Rel
  by blast
moreover assume SP ↓ <SWB> a
hence SourceTerm SP ↓.a
  by simp
ultimately have SourceTerm SQ ↓.a
  using preservation preservation-of-barbs-in-barbed-encoding[where Rel=Rel]
  by blast
thus SQ ↓ <SWB> a
  by simp
qed

```

**lemma** (in *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-preserve-barbs*:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes preservation: rel-preserves-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
shows rel-preserves-barbs SRel SWB
  and rel-preserves-barbs TRel TWB

```

**proof** –

```

show rel-preserves-barbs SRel SWB
  using preservation rel-with-source-impl-SRel-preserves-barbs[where
    Rel=indRelRSTPO SRel TRel and SRel=SRel]
  by (simp add: indRelRSTPO.source)
next
show rel-preserves-barbs TRel TWB
  using preservation rel-with-target-impl-TRel-preserves-barbs[where
    Rel=indRelRSTPO SRel TRel and TRel=TRel]
  by (simp add: indRelRSTPO.target)
qed

```

**lemma** (in *encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-preserve-barbs*:

```

fixes SRel :: ('procS × 'procS) set

```

```

    and TRel :: ('procT × 'procT) set
  assumes preservation: rel-preserves-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-preserves-barbs SRel SWB
    and rel-preserves-barbs TRel TWB
proof -
  show rel-preserves-barbs SRel SWB
    using preservation rel-with-source-impl-SRel-preserves-barbs[where
      Rel=indRelLSTPO SRel TRel and SRel=SRel]
    by (simp add: indRelLSTPO.source)
next
  show rel-preserves-barbs TRel TWB
    using preservation rel-with-target-impl-TRel-preserves-barbs[where
      Rel=indRelLSTPO SRel TRel and TRel=TRel]
    by (simp add: indRelLSTPO.target)
qed

```

```

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-preserve-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes preservation: rel-preserves-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  shows rel-preserves-barbs SRel SWB
    and rel-preserves-barbs TRel TWB
proof -
  show rel-preserves-barbs SRel SWB
    using preservation rel-with-source-impl-SRel-preserves-barbs[where
      Rel=indRelSTEQ SRel TRel and SRel=SRel]
    by (simp add: indRelSTEQ.source)
next
  show rel-preserves-barbs TRel TWB
    using preservation rel-with-target-impl-TRel-preserves-barbs[where
      Rel=indRelSTEQ SRel TRel and TRel=TRel]
    by (simp add: indRelSTEQ.target)
qed

```

```

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-weakly-preserves-barbs:
  fixes SRel :: ('procS × 'procS) set
    and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)
    and sourceInRel: ∀ S1 S2. (S1, S2) ∈ SRel → (SourceTerm S1, SourceTerm S2) ∈ Rel
  shows rel-weakly-preserves-barbs SRel SWB
proof clarify
  fix SP SQ a SP'
  assume (SP, SQ) ∈ SRel
  with sourceInRel have (SourceTerm SP, SourceTerm SQ) ∈ Rel
    by blast
  moreover assume SP ↦ (Calculus SWB)* SP' and SP' ↓ <SWB> a
  hence SourceTerm SP ↓.a
    by blast
  ultimately have SourceTerm SQ ↓.a
    using preservation weak-preservation-of-barbs-in-barbed-encoding[where Rel=Rel]
    by blast
  thus SQ ↓ <SWB> a
    by simp
qed

```

```

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-barbs SRel SWB
    and rel-weakly-preserves-barbs TRel TWB

```

```

proof –
  show rel-weakly-preserves-barbs SRel SWB
    using preservation rel-with-source-impl-SRel-weakly-preserves-barbs [where
      Rel=indRelRSTPO SRel TRel and SRel=SRel]
    by (simp add: indRelRSTPO.source)
next
  show rel-weakly-preserves-barbs TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-barbs [where
      Rel=indRelRSTPO SRel TRel and TRel=TRel]
    by (simp add: indRelRSTPO.target)
qed

```

```

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-barbs SRel SWB
    and rel-weakly-preserves-barbs TRel TWB

```

```

proof –
  show rel-weakly-preserves-barbs SRel SWB
    using preservation rel-with-source-impl-SRel-weakly-preserves-barbs [where
      Rel=indRelLSTPO SRel TRel and SRel=SRel]
    by (simp add: indRelLSTPO.source)
next
  show rel-weakly-preserves-barbs TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-barbs [where
      Rel=indRelLSTPO SRel TRel and TRel=TRel]
    by (simp add: indRelLSTPO.target)
qed

```

```

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-barbs SRel SWB
    and rel-weakly-preserves-barbs TRel TWB

```

```

proof –
  show rel-weakly-preserves-barbs SRel SWB
    using preservation rel-with-source-impl-SRel-weakly-preserves-barbs [where
      Rel=indRelSTEQ SRel TRel and SRel=SRel]
    by (simp add: indRelSTEQ.source)
next
  show rel-weakly-preserves-barbs TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-barbs [where
      Rel=indRelSTEQ SRel TRel and TRel=TRel]
    by (simp add: indRelSTEQ.target)
qed

```

If *indRelRSTPO*, *indRelLSTPO*, or *indRelSTPO* reflects barbs then so do the corresponding source term and target term relations.

```

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-reflects-barbs:
  fixes SRel :: ('procS × 'procS) set
    and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)
    and sourceInRel: ∀ S1 S2. (S1, S2) ∈ SRel → (SourceTerm S1, SourceTerm S2) ∈ Rel
  shows rel-reflects-barbs SRel SWB

```

```

proof clarify
  fix SP SQ a
  assume (SP, SQ) ∈ SRel
  with sourceInRel have (SourceTerm SP, SourceTerm SQ) ∈ Rel

```

by *blast*  
 moreover assume  $SQ \downarrow < SWB > a$   
 hence *SourceTerm*  $SQ \downarrow . a$   
 by *simp*  
 ultimately have *SourceTerm*  $SP \downarrow . a$   
 using *reflection reflection-of-barbs-in-barbed-encoding*[**where**  $Rel = Rel$ ]  
 by *blast*  
 thus  $SP \downarrow < SWB > a$   
 by *simp*  
 qed

**lemma** (*in encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-reflect-barbs*:  
 fixes  $SRel :: ('procS \times 'procS) \text{ set}$   
 and  $TRel :: ('procT \times 'procT) \text{ set}$   
 assumes *reflection: rel-reflects-barbs* (*indRelRSTPO*  $SRel$   $TRel$ ) (*STCalWB SWB TWB*)  
 shows *rel-reflects-barbs*  $SRel$   $SWB$   
 and *rel-reflects-barbs*  $TRel$   $TWB$   
**proof** –  
 show *rel-reflects-barbs*  $SRel$   $SWB$   
 using *reflection rel-with-source-impl-SRel-reflects-barbs*[**where**  
 $Rel = indRelRSTPO$   $SRel$   $TRel$  **and**  $SRel = SRel$ ]  
 by (*simp add: indRelRSTPO.source*)  
**next**  
 show *rel-reflects-barbs*  $TRel$   $TWB$   
 using *reflection rel-with-target-impl-TRel-reflects-barbs*[**where**  
 $Rel = indRelRSTPO$   $SRel$   $TRel$  **and**  $TRel = TRel$ ]  
 by (*simp add: indRelRSTPO.target*)  
 qed

**lemma** (*in encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-reflect-barbs*:  
 fixes  $SRel :: ('procS \times 'procS) \text{ set}$   
 and  $TRel :: ('procT \times 'procT) \text{ set}$   
 assumes *reflection: rel-reflects-barbs* (*indRelLSTPO*  $SRel$   $TRel$ ) (*STCalWB SWB TWB*)  
 shows *rel-reflects-barbs*  $SRel$   $SWB$   
 and *rel-reflects-barbs*  $TRel$   $TWB$   
**proof** –  
 show *rel-reflects-barbs*  $SRel$   $SWB$   
 using *reflection rel-with-source-impl-SRel-reflects-barbs*[**where**  
 $Rel = indRelLSTPO$   $SRel$   $TRel$  **and**  $SRel = SRel$ ]  
 by (*simp add: indRelLSTPO.source*)  
**next**  
 show *rel-reflects-barbs*  $TRel$   $TWB$   
 using *reflection rel-with-target-impl-TRel-reflects-barbs*[**where**  
 $Rel = indRelLSTPO$   $SRel$   $TRel$  **and**  $TRel = TRel$ ]  
 by (*simp add: indRelLSTPO.target*)  
 qed

**lemma** (*in encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-reflect-barbs*:  
 fixes  $SRel :: ('procS \times 'procS) \text{ set}$   
 and  $TRel :: ('procT \times 'procT) \text{ set}$   
 assumes *reflection: rel-reflects-barbs* (*indRelSTEQ*  $SRel$   $TRel$ ) (*STCalWB SWB TWB*)  
 shows *rel-reflects-barbs*  $SRel$   $SWB$   
 and *rel-reflects-barbs*  $TRel$   $TWB$   
**proof** –  
 show *rel-reflects-barbs*  $SRel$   $SWB$   
 using *reflection rel-with-source-impl-SRel-reflects-barbs*[**where**  
 $Rel = indRelSTEQ$   $SRel$   $TRel$  **and**  $SRel = SRel$ ]  
 by (*simp add: indRelSTEQ.source*)  
**next**  
 show *rel-reflects-barbs*  $TRel$   $TWB$   
 using *reflection rel-with-target-impl-TRel-reflects-barbs*[**where**

$Rel = indRelSTEQ\ SRel\ TRel\ \mathbf{and}\ TRel = TRel]$

**by** (*simp add: indRelSTEQ.target*)  
**qed**

**lemma** (*in encoding-wrt-barbs*) *rel-with-source-impl-SRel-weakly-reflects-barbs*:  
**fixes**  $SRel :: ('procS \times 'procS)\ set$   
**and**  $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$   
**assumes** *reflection: rel-weakly-reflects-barbs Rel (STCalWB SWB TWB)*  
**and** *sourceInRel:  $\forall S1\ S2. (S1, S2) \in SRel \longrightarrow (SourceTerm\ S1, SourceTerm\ S2) \in Rel$*   
**shows** *rel-weakly-reflects-barbs SRel SWB*

**proof** *clarify*  
**fix**  $SP\ SQ\ a\ SQ'$   
**assume**  $(SP, SQ) \in SRel$   
**with** *sourceInRel* **have**  $(SourceTerm\ SP, SourceTerm\ SQ) \in Rel$   
**by** *blast*  
**moreover** **assume**  $SQ \longmapsto (Calculus\ SWB)^* SQ'$  **and**  $SQ \Downarrow_{SWB} a$   
**hence**  $SourceTerm\ SQ \Downarrow a$   
**by** *blast*  
**ultimately** **have**  $SourceTerm\ SP \Downarrow a$   
**using** *reflection weak-reflection-of-barbs-in-barbed-encoding* [**where**  $Rel = Rel$ ]  
**by** *blast*  
**thus**  $SP \Downarrow_{SWB} a$   
**by** *simp*  
**qed**

**lemma** (*in encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs*:  
**fixes**  $SRel :: ('procS \times 'procS)\ set$   
**and**  $TRel :: ('procT \times 'procT)\ set$   
**assumes** *reflection: rel-weakly-reflects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)*  
**shows** *rel-weakly-reflects-barbs SRel SWB*  
**and** *rel-weakly-reflects-barbs TRel TWB*

**proof** –  
**show** *rel-weakly-reflects-barbs SRel SWB*  
**using** *reflection rel-with-source-impl-SRel-weakly-reflects-barbs* [**where**  
 $Rel = indRelRSTPO\ SRel\ TRel\ \mathbf{and}\ SRel = SRel]$   
**by** (*simp add: indRelRSTPO.source*)  
**next**  
**show** *rel-weakly-reflects-barbs TRel TWB*  
**using** *reflection rel-with-target-impl-TRel-weakly-reflects-barbs* [**where**  
 $Rel = indRelRSTPO\ SRel\ TRel\ \mathbf{and}\ TRel = TRel]$   
**by** (*simp add: indRelRSTPO.target*)  
**qed**

**lemma** (*in encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs*:  
**fixes**  $SRel :: ('procS \times 'procS)\ set$   
**and**  $TRel :: ('procT \times 'procT)\ set$   
**assumes** *reflection: rel-weakly-reflects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)*  
**shows** *rel-weakly-reflects-barbs SRel SWB*  
**and** *rel-weakly-reflects-barbs TRel TWB*

**proof** –  
**show** *rel-weakly-reflects-barbs SRel SWB*  
**using** *reflection rel-with-source-impl-SRel-weakly-reflects-barbs* [**where**  
 $Rel = indRelLSTPO\ SRel\ TRel\ \mathbf{and}\ SRel = SRel]$   
**by** (*simp add: indRelLSTPO.source*)  
**next**  
**show** *rel-weakly-reflects-barbs TRel TWB*  
**using** *reflection rel-with-target-impl-TRel-weakly-reflects-barbs* [**where**  
 $Rel = indRelLSTPO\ SRel\ TRel\ \mathbf{and}\ TRel = TRel]$   
**by** (*simp add: indRelLSTPO.target*)  
**qed**



```

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes reflection: rel-weakly-reflects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-reflects-barbs SRel SWB
    and rel-weakly-reflects-barbs TRel TWB
proof –
  show rel-weakly-reflects-barbs SRel SWB
    using reflection rel-with-source-impl-SRel-weakly-reflects-barbs[where
      Rel=indRelSTEQ SRel TRel and SRel=SRel]
    by (simp add: indRelSTEQ.source)
next
  show rel-weakly-reflects-barbs TRel TWB
    using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where
      Rel=indRelSTEQ SRel TRel and TRel=TRel]
    by (simp add: indRelSTEQ.target)
qed

```

If *indRelRSTPO*, *indRelLSTPO*, or *indRelSTPO* respects barbs then so do the corresponding source term and target term relations.

```

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-respect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes respection: rel-respects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs SRel SWB
    and rel-respects-barbs TRel TWB
proof –
  show rel-respects-barbs SRel SWB
    using respection
      indRelRSTPO-impl-SRel-and-TRel-preserve-barbs(1)[where SRel=SRel and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-reflect-barbs(1)[where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
      indRelRSTPO-impl-SRel-and-TRel-preserve-barbs(2)[where SRel=SRel and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-reflect-barbs(2)[where SRel=SRel and TRel=TRel]
    by blast
qed

```

```

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-respect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes respection: rel-respects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs SRel SWB
    and rel-respects-barbs TRel TWB
proof –
  show rel-respects-barbs SRel SWB
    using respection
      indRelLSTPO-impl-SRel-and-TRel-preserve-barbs(1)[where SRel=SRel and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-reflect-barbs(1)[where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
      indRelLSTPO-impl-SRel-and-TRel-preserve-barbs(2)[where SRel=SRel and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-reflect-barbs(2)[where SRel=SRel and TRel=TRel]
    by blast
qed

```

```

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-respect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes respection: rel-respects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs SRel SWB
    and rel-respects-barbs TRel TWB
proof –
  show rel-respects-barbs SRel SWB
    using respection
      indRelSTEQ-impl-SRel-and-TRel-preserve-barbs(1)[where SRel=SRel and TRel=TRel]
      indRelSTEQ-impl-SRel-and-TRel-reflect-barbs(1)[where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
      indRelSTEQ-impl-SRel-and-TRel-preserve-barbs(2)[where SRel=SRel and TRel=TRel]
      indRelSTEQ-impl-SRel-and-TRel-reflect-barbs(2)[where SRel=SRel and TRel=TRel]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes respection: rel-weakly-respects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-respects-barbs SRel SWB
    and rel-weakly-respects-barbs TRel TWB
proof –
  show rel-weakly-respects-barbs SRel SWB
    using respection indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(1)[where SRel=SRel
      and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(1)[where SRel=SRel
      and TRel=TRel]
    by blast
next
  show rel-weakly-respects-barbs TRel TWB
    using respection indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(2)[where SRel=SRel
      and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(2)[where SRel=SRel
      and TRel=TRel]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes respection: rel-weakly-respects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-respects-barbs SRel SWB
    and rel-weakly-respects-barbs TRel TWB
proof –
  show rel-weakly-respects-barbs SRel SWB
    using respection indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(1)[where SRel=SRel
      and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(1)[where SRel=SRel
      and TRel=TRel]
    by blast
next
  show rel-weakly-respects-barbs TRel TWB
    using respection indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(2)[where SRel=SRel
      and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(2)[where SRel=SRel
      and TRel=TRel]

```

by *blast*  
qed

**lemma** (in *encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-weakly-respect-barbs*:  
**fixes**  $SRel :: ('procS \times 'procS) \text{ set}$   
**and**  $TRel :: ('procT \times 'procT) \text{ set}$   
**assumes** *respection: rel-weakly-respects-barbs* (*indRelSTEQ*  $SRel$   $TRel$ ) (*STCalWB SWB TWB*)  
**shows** *rel-weakly-respects-barbs*  $SRel$   $SWB$   
**and** *rel-weakly-respects-barbs*  $TRel$   $TWB$   
**proof** –  
**show** *rel-weakly-respects-barbs*  $SRel$   $SWB$   
**using** *respection indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs(1)*[**where**  $SRel=SRel$   
**and**  $TRel=TRel$ ]  
*indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs(1)*[**where**  $SRel=SRel$   
**and**  $TRel=TRel$ ]  
by *blast*  
**next**  
**show** *rel-weakly-respects-barbs*  $TRel$   $TWB$   
**using** *respection indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs(2)*[**where**  $SRel=SRel$   
**and**  $TRel=TRel$ ]  
*indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs(2)*[**where**  $SRel=SRel$   
**and**  $TRel=TRel$ ]  
by *blast*  
qed

If  $TRel$  is reflexive then  $\text{ind relRTPO}$  is a subrelation of  $\text{indRelTEQ}$ . If  $SRel$  is reflexive then  $\text{indRelRTPO}$  is a subrelation of  $\text{indRelRTPO}$ . Moreover,  $\text{indRelRSTPO}$  is a subrelation of  $\text{indRelSTEQ}$ .

**lemma** (in *encoding*) *indRelRTPO-to-indRelTEQ*:  
**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $P Q :: ('procS, 'procT) \text{ Proc}$   
**assumes** *rel*:  $P \lesssim[\cdot] RT < TRel > Q$   
**and** *reflT*: *refl*  $TRel$   
**shows**  $P \sim[\cdot] T < TRel > Q$   
**using** *rel*  
**proof** *induct*  
**case** (*encR*  $S$ )  
**show** *SourceTerm*  $S \sim[\cdot] T < TRel > \text{TargetTerm } (\llbracket S \rrbracket)$   
**by** (*rule indRelTEQ.encR*)  
**next**  
**case** (*source*  $S$ )  
**from** *reflT* **show** *SourceTerm*  $S \sim[\cdot] T < TRel > \text{SourceTerm } S$   
**using** *indRelTEQ-refl*[of  $TRel$ ]  
**unfolding** *refl-on-def*  
**by** *simp*  
**next**  
**case** (*target*  $T1$   $T2$ )  
**assume**  $(T1, T2) \in TRel$   
**thus** *TargetTerm*  $T1 \sim[\cdot] T < TRel > \text{TargetTerm } T2$   
**by** (*rule indRelTEQ.target*)  
**next**  
**case** (*trans*  $TP$   $TQ$   $TR$ )  
**assume**  $TP \sim[\cdot] T < TRel > TQ$  **and**  $TQ \sim[\cdot] T < TRel > TR$   
**thus**  $TP \sim[\cdot] T < TRel > TR$   
**by** (*rule indRelTEQ.trans*)  
qed

**lemma** (in *encoding*) *indRelRTPO-to-indRelRSTPO*:  
**fixes**  $SRel :: ('procS \times 'procS) \text{ set}$   
**and**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $P Q :: ('procS, 'procT) \text{ Proc}$

```

assumes rel:  $P \lesssim[\cdot]RT \langle TRel \rangle Q$ 
and reflS: refl SRel
shows  $P \lesssim[\cdot]R \langle SRel, TRel \rangle Q$ 
using rel
proof induct
case (encR S)
show  $SourceTerm\ S \lesssim[\cdot]R \langle SRel, TRel \rangle TargetTerm\ ([S])$ 
by (rule indRelRSTPO.encR)
next
case (source S)
from reflS show  $SourceTerm\ S \lesssim[\cdot]R \langle SRel, TRel \rangle SourceTerm\ S$ 
unfolding refl-on-def
by (simp add: indRelRSTPO.source)
next
case (target T1 T2)
assume  $(T1, T2) \in TRel$ 
thus  $TargetTerm\ T1 \lesssim[\cdot]R \langle SRel, TRel \rangle TargetTerm\ T2$ 
by (rule indRelRSTPO.target)
next
case (trans P Q R)
assume  $P \lesssim[\cdot]R \langle SRel, TRel \rangle Q$  and  $Q \lesssim[\cdot]R \langle SRel, TRel \rangle R$ 
thus  $P \lesssim[\cdot]R \langle SRel, TRel \rangle R$ 
by (rule indRelRSTPO.trans)
qed

```

```

lemma (in encoding) indRelRSTPO-to-indRelSTEQ:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and P Q :: ('procS, 'procT) Proc
assumes rel:  $P \lesssim[\cdot]R \langle SRel, TRel \rangle Q$ 
shows  $P \sim[\cdot] \langle SRel, TRel \rangle Q$ 
using rel
proof induct
case (encR S)
show  $SourceTerm\ S \sim[\cdot] \langle SRel, TRel \rangle TargetTerm\ ([S])$ 
by (rule indRelSTEQ.encR)
next
case (source S1 S2)
assume  $(S1, S2) \in SRel$ 
thus  $SourceTerm\ S1 \sim[\cdot] \langle SRel, TRel \rangle SourceTerm\ S2$ 
by (rule indRelSTEQ.source)
next
case (target T1 T2)
assume  $(T1, T2) \in TRel$ 
thus  $TargetTerm\ T1 \sim[\cdot] \langle SRel, TRel \rangle TargetTerm\ T2$ 
by (rule indRelSTEQ.target)
next
case (trans P Q R)
assume  $P \sim[\cdot] \langle SRel, TRel \rangle Q$  and  $Q \sim[\cdot] \langle SRel, TRel \rangle R$ 
thus  $P \sim[\cdot] \langle SRel, TRel \rangle R$ 
by (rule indRelSTEQ.trans)
qed

```

If *indRelRTPO* is a bisimulation and *SRel* is a reflexive bisimulation then also *indRelRSTPO* is a bisimulation.

```

lemma (in encoding) indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes bisimT: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
and bisimS: weak-reduction-bisimulation SRel Source

```

**and** *reflS*: *refl SRel*  
**shows** *weak-reduction-bisimulation* (*indRelRSTPO SRel TRel*) (*STCal Source Target*)  
**proof** *auto*  
**fix**  $P Q P'$   
**assume**  $P \lesssim[\cdot]R\langle SRel, TRel \rangle Q$  **and**  $P \mapsto (STCal\ Source\ Target)* P'$   
**thus**  $\exists Q'. Q \mapsto (STCal\ Source\ Target)* Q' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**proof** (*induct arbitrary: P'*)  
**case** (*encR S*)  
**have** *SourceTerm S*  $\lesssim[\cdot]RT\langle TRel \rangle TargetTerm ([S])$   
**by** (*rule indRelRTPO.encR*)  
**moreover assume** *SourceTerm S*  $\mapsto (STCal\ Source\ Target)* P'$   
**ultimately obtain**  $Q'$  **where**  $A1: TargetTerm ([S]) \mapsto (STCal\ Source\ Target)* Q'$   
**and**  $A2: P' \lesssim[\cdot]RT\langle TRel \rangle Q'$   
**using** *bisimT*  
**by** *blast*  
**from** *reflS A2* **have**  $P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** (*simp add: indRelRTPO-to-indRelRSTPO*)  
**with**  $A1$  **show**  $\exists Q'. TargetTerm ([S]) \mapsto (STCal\ Source\ Target)* Q' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** *blast*  
**next**  
**case** (*source S1 S2*)  
**assume** *SourceTerm S1*  $\mapsto (STCal\ Source\ Target)* P'$   
**from** *this* **obtain**  $S1'$  **where**  $B1: S1' \in S P'$  **and**  $B2: S1 \mapsto Source* S1'$   
**by** (*auto simp add: STCal-steps(1)*)  
**assume**  $(S1, S2) \in SRel$   
**with**  $B2$  *bisimS* **obtain**  $S2'$  **where**  $B3: S2 \mapsto Source* S2'$  **and**  $B4: (S1', S2') \in SRel$   
**by** *blast*  
**from**  $B3$  **have** *SourceTerm S2*  $\mapsto (STCal\ Source\ Target)* (SourceTerm S2')$   
**by** (*simp add: STCal-steps(1)*)  
**moreover from**  $B1 B4$  **have**  $P' \lesssim[\cdot]R\langle SRel, TRel \rangle SourceTerm S2'$   
**by** (*simp add: indRelRSTPO.source*)  
**ultimately show**  $\exists Q'. SourceTerm S2 \mapsto (STCal\ Source\ Target)* Q' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $(T1, T2) \in TRel$   
**hence** *TargetTerm T1*  $\lesssim[\cdot]RT\langle TRel \rangle TargetTerm T2$   
**by** (*rule indRelRTPO.target*)  
**moreover assume** *TargetTerm T1*  $\mapsto (STCal\ Source\ Target)* P'$   
**ultimately obtain**  $Q'$  **where**  $C1: TargetTerm T2 \mapsto (STCal\ Source\ Target)* Q'$   
**and**  $C2: P' \lesssim[\cdot]RT\langle TRel \rangle Q'$   
**using** *bisimT*  
**by** *blast*  
**from** *reflS C2* **have**  $P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** (*simp add: indRelRTPO-to-indRelRSTPO*)  
**with**  $C1$  **show**  $\exists Q'. TargetTerm T2 \mapsto (STCal\ Source\ Target)* Q' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** *blast*  
**next**  
**case** (*trans P Q R*)  
**assume**  $P \mapsto (STCal\ Source\ Target)* P'$   
**and**  $\bigwedge P'. P \mapsto (STCal\ Source\ Target)* P'$   
 $\implies \exists Q'. Q \mapsto (STCal\ Source\ Target)* Q' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**from** *this* **obtain**  $Q'$  **where**  $D1: Q \mapsto (STCal\ Source\ Target)* Q'$  **and**  $D2: P' \lesssim[\cdot]R\langle SRel, TRel \rangle Q'$   
**by** *blast*  
**assume**  $\bigwedge Q'. Q \mapsto (STCal\ Source\ Target)* Q'$   
 $\implies \exists R'. R \mapsto (STCal\ Source\ Target)* R' \wedge Q' \lesssim[\cdot]R\langle SRel, TRel \rangle R'$   
**with**  $D1$  **obtain**  $R'$  **where**  $D3: R \mapsto (STCal\ Source\ Target)* R'$  **and**  $D4: Q' \lesssim[\cdot]R\langle SRel, TRel \rangle R'$   
**by** *blast*  
**from**  $D2 D4$  **have**  $P' \lesssim[\cdot]R\langle SRel, TRel \rangle R'$   
**by** (*rule indRelRSTPO.trans*)  
**with**  $D3$  **show**  $\exists R'. R \mapsto (STCal\ Source\ Target)* R' \wedge P' \lesssim[\cdot]R\langle SRel, TRel \rangle R'$

by *blast*  
 qed  
 next  
 fix  $P Q Q'$   
 assume  $P \lesssim [\cdot]R \langle SRel, TRel \rangle Q$  and  $Q \mapsto (STCal \text{ Source Target})^* Q'$   
 thus  $\exists P'. P \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 proof (induct arbitrary:  $Q'$ )  
 case (encR  $S$ )  
 have  $SourceTerm S \lesssim [\cdot]RT \langle TRel \rangle TargetTerm ([S])$   
 by (rule indRelRTPO.encR)  
 moreover assume  $TargetTerm ([S]) \mapsto (STCal \text{ Source Target})^* Q'$   
 ultimately obtain  $P'$  where  $E1: SourceTerm S \mapsto (STCal \text{ Source Target})^* P'$   
 and  $E2: P' \lesssim [\cdot]RT \langle TRel \rangle Q'$   
 using *bisimT*  
 by *blast*  
 from *reflS*  $E2$  have  $P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by (*simp add: indRelRTPO-to-indRelRSTPO*)  
 with  $E1$  show  $\exists P'. SourceTerm S \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by *blast*  
 next  
 case (source  $S1 S2$ )  
 assume  $SourceTerm S2 \mapsto (STCal \text{ Source Target})^* Q'$   
 from *this* obtain  $S2'$  where  $F1: S2' \in S Q'$  and  $F2: S2 \mapsto Source^* S2'$   
 by (*auto simp add: STCal-steps(1)*)  
 assume  $(S1, S2) \in SRel$   
 with  $F2$  *bisimS* obtain  $S1'$  where  $F3: S1 \mapsto Source^* S1'$  and  $F4: (S1', S2') \in SRel$   
 by *blast*  
 from  $F3$  have  $SourceTerm S1 \mapsto (STCal \text{ Source Target})^* (SourceTerm S1')$   
 by (*simp add: STCal-steps(1)*)  
 moreover from  $F1 F4$  have  $SourceTerm S1' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by (*simp add: indRelRSTPO.source*)  
 ultimately show  $\exists P'. SourceTerm S1 \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by *blast*  
 next  
 case (target  $T1 T2$ )  
 assume  $(T1, T2) \in TRel$   
 hence  $TargetTerm T1 \lesssim [\cdot]RT \langle TRel \rangle TargetTerm T2$   
 by (rule indRelRTPO.target)  
 moreover assume  $TargetTerm T2 \mapsto (STCal \text{ Source Target})^* Q'$   
 ultimately obtain  $P'$  where  $G1: TargetTerm T1 \mapsto (STCal \text{ Source Target})^* P'$   
 and  $G2: P' \lesssim [\cdot]RT \langle TRel \rangle Q'$   
 using *bisimT*  
 by *blast*  
 from *reflS*  $G2$  have  $P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by (*simp add: indRelRTPO-to-indRelRSTPO*)  
 with  $G1$  show  $\exists P'. TargetTerm T1 \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by *blast*  
 next  
 case (trans  $P Q R R'$ )  
 assume  $R \mapsto (STCal \text{ Source Target})^* R'$   
 and  $\bigwedge R'. R \mapsto (STCal \text{ Source Target})^* R'$   
 $\implies \exists Q'. Q \mapsto (STCal \text{ Source Target})^* Q' \wedge Q' \lesssim [\cdot]R \langle SRel, TRel \rangle R'$   
 from *this* obtain  $Q'$  where  $H1: Q \mapsto (STCal \text{ Source Target})^* Q'$  and  $H2: Q' \lesssim [\cdot]R \langle SRel, TRel \rangle R'$   
 by *blast*  
 assume  $\bigwedge Q'. Q \mapsto (STCal \text{ Source Target})^* Q'$   
 $\implies \exists P'. P \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 with  $H1$  obtain  $P'$  where  $H3: P \mapsto (STCal \text{ Source Target})^* P'$  and  $H4: P' \lesssim [\cdot]R \langle SRel, TRel \rangle Q'$   
 by *blast*  
 from  $H4 H2$  have  $P' \lesssim [\cdot]R \langle SRel, TRel \rangle R'$   
 by (rule indRelRSTPO.trans)  
 with  $H3$  show  $\exists P'. P \mapsto (STCal \text{ Source Target})^* P' \wedge P' \lesssim [\cdot]R \langle SRel, TRel \rangle R'$

```

    by blast
qed
qed

end
theory SuccessSensitiveness
  imports SourceTargetRelation
begin

```

## 6 Success Sensitiveness and Barbs

To compare the abstract behavior of two terms, often some notion of success or successful termination is used. Daniele Gorla assumes a constant process (similar to the empty process) that represents successful termination in order to compare the behavior of source terms with their literal translations. Then an encoding is success sensitive if, for all source terms  $S$ ,  $S$  reaches success iff the translation of  $S$  reaches success. Successful termination can be considered as some special kind of barb. Accordingly we generalize successful termination to the respectation of an arbitrary subset of barbs. An encoding respects a set of barbs if, for every source term  $S$  and all considered barbs  $a$ ,  $S$  reaches  $a$  iff the translation of  $S$  reaches  $a$ .

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-preserves-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-weakly-preserves-barb-set* Barbs  $\equiv$  *enc-preserves-binary-pred* ( $\lambda P a. a \in \text{Barbs} \wedge P \Downarrow . a$ )

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-preserves-barbs* :: bool **where**  
*enc-weakly-preserves-barbs*  $\equiv$  *enc-preserves-binary-pred* ( $\lambda P a. P \Downarrow . a$ )

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-preserves-barbs-and-barb-set*:  
**shows** *enc-weakly-preserves-barbs* = ( $\forall$  Barbs. *enc-weakly-preserves-barb-set* Barbs)  
**by** *blast*

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-reflects-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-weakly-reflects-barb-set* Barbs  $\equiv$  *enc-reflects-binary-pred* ( $\lambda P a. a \in \text{Barbs} \wedge P \Downarrow . a$ )

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-reflects-barbs* :: bool **where**  
*enc-weakly-reflects-barbs*  $\equiv$  *enc-reflects-binary-pred* ( $\lambda P a. P \Downarrow . a$ )

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-reflects-barbs-and-barb-set*:  
**shows** *enc-weakly-reflects-barbs* = ( $\forall$  Barbs. *enc-weakly-reflects-barb-set* Barbs)  
**by** *blast*

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-respects-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-weakly-respects-barb-set* Barbs  $\equiv$   
*enc-weakly-preserves-barb-set* Barbs  $\wedge$  *enc-weakly-reflects-barb-set* Barbs

**abbreviation** (in *encoding-wrt-barbs*) *enc-weakly-respects-barbs* :: bool **where**  
*enc-weakly-respects-barbs*  $\equiv$  *enc-weakly-preserves-barbs*  $\wedge$  *enc-weakly-reflects-barbs*

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-respects-barbs-and-barb-set*:  
**shows** *enc-weakly-respects-barbs* = ( $\forall$  Barbs. *enc-weakly-respects-barb-set* Barbs)

**proof** –

**have** ( $\forall$  Barbs. *enc-weakly-respects-barb-set* Barbs)  
= ( $\forall$  Barbs. ( $\forall S x. x \in \text{Barbs} \wedge S \Downarrow \langle \text{SWB} \rangle x \longrightarrow \llbracket S \rrbracket \Downarrow \langle \text{TWB} \rangle x$ )  
 $\wedge$  ( $\forall S x. x \in \text{Barbs} \wedge \llbracket S \rrbracket \Downarrow \langle \text{TWB} \rangle x \longrightarrow S \Downarrow \langle \text{SWB} \rangle x$ ))

**by** *simp*

**hence** ( $\forall$  Barbs. *enc-weakly-respects-barb-set* Barbs)  
= (( $\forall$  Barbs. *enc-weakly-preserves-barb-set* Barbs)  
 $\wedge$  ( $\forall$  Barbs. *enc-weakly-reflects-barb-set* Barbs))

**apply** *simp* **by** *fast*

**thus** *?thesis*

**apply simp by blast**  
**qed**

An encoding strongly respects some set of barbs if, for every source term  $S$  and all considered barbs  $a$ ,  $S$  has  $a$  iff the translation of  $S$  has  $a$ .

**abbreviation** (in *encoding-wrt-barbs*) *enc-preserves-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-preserves-barb-set*  $Barbs \equiv enc-preserves-binary-pred (\lambda P a. a \in Barbs \wedge P \downarrow . a)$

**abbreviation** (in *encoding-wrt-barbs*) *enc-preserves-barbs* :: bool **where**  
*enc-preserves-barbs*  $\equiv enc-preserves-binary-pred (\lambda P a. P \downarrow . a)$

**lemma** (in *encoding-wrt-barbs*) *enc-preserves-barbs-and-barb-set*:  
**shows** *enc-preserves-barbs* =  $(\forall Barbs. enc-preserves-barb-set Barbs)$   
**by** *blast*

**abbreviation** (in *encoding-wrt-barbs*) *enc-reflects-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-reflects-barb-set*  $Barbs \equiv enc-reflects-binary-pred (\lambda P a. a \in Barbs \wedge P \downarrow . a)$

**abbreviation** (in *encoding-wrt-barbs*) *enc-reflects-barbs* :: bool **where**  
*enc-reflects-barbs*  $\equiv enc-reflects-binary-pred (\lambda P a. P \downarrow . a)$

**lemma** (in *encoding-wrt-barbs*) *enc-reflects-barbs-and-barb-set*:  
**shows** *enc-reflects-barbs* =  $(\forall Barbs. enc-reflects-barb-set Barbs)$   
**by** *blast*

**abbreviation** (in *encoding-wrt-barbs*) *enc-respects-barb-set* :: 'barbs set  $\Rightarrow$  bool **where**  
*enc-respects-barb-set*  $Barbs \equiv enc-preserves-barb-set Barbs \wedge enc-reflects-barb-set Barbs$

**abbreviation** (in *encoding-wrt-barbs*) *enc-respects-barbs* :: bool **where**  
*enc-respects-barbs*  $\equiv enc-preserves-barbs \wedge enc-reflects-barbs$

**lemma** (in *encoding-wrt-barbs*) *enc-respects-barbs-and-barb-set*:  
**shows** *enc-respects-barbs* =  $(\forall Barbs. enc-respects-barb-set Barbs)$

**proof** –

**have**  $(\forall Barbs. enc-respects-barb-set Barbs)$   
=  $((\forall Barbs. enc-preserves-barb-set Barbs)$   
 $\wedge (\forall Barbs. enc-reflects-barb-set Barbs))$

**apply simp by fast**

**thus** *?thesis*

**apply simp by blast**

**qed**

An encoding (weakly) preserves barbs iff (1) there exists a relation, like *indRelR*, that relates source terms and their literal translations and preserves (reachability/existence of barbs, or (2) there exists a relation, like *indRelL*, that relates literal translations and their source terms and reflects (reachability/existence of barbs.

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-preserves-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**and** *TRel* :: ('procT  $\times$  'procT) set

**shows** *enc-weakly-preserves-barb-set Barbs*

=  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-weakly-preserves-barb-set Rel (STCalWB SWB TWB) Barbs)$

**using** *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred* [**where**

$Pred = \lambda P a. a \in Barbs \wedge P \downarrow < STCalWB SWB TWB > a]$  *STCalWB-reachesBarbST*

**by** *simp*

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-preserves-barbs-iff-source-target-rel*:

**shows** *enc-weakly-preserves-barbs*

=  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-weakly-preserves-barbs Rel (STCalWB SWB TWB))$



**using** *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**  
*Pred*= $\lambda P a. P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-reachesBarbST*  
**by** *simp*

**lemma** (**in** *encoding-wrt-barbs*) *enc-preserves-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**shows** *enc-preserves-barb-set Barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-preserves-barb-set } Rel (STCalWB SWB TWB) Barbs)$

**using** *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**  
*Pred*= $\lambda P a. a \in Barbs \wedge P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

**lemma** (**in** *encoding-wrt-barbs*) *enc-preserves-barbs-iff-source-target-rel*:

**shows** *enc-preserves-barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-preserves-barbs } Rel (STCalWB SWB TWB)$ )

**using** *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**  
*Pred*= $\lambda P a. P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

An encoding (weakly) reflects barbs iff (1) there exists a relation, like *indRelR*, that relates source terms and their literal translations and reflects (reachability/existence of barbs, or (2) there exists a relation, like *indRelL*, that relates literal translations and their source terms and preserves (reachability/existence of barbs.

**lemma** (**in** *encoding-wrt-barbs*) *enc-weakly-reflects-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**shows** *enc-weakly-reflects-barb-set Barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-weakly-reflects-barb-set } Rel (STCalWB SWB TWB) Barbs)$

**using** *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**  
*Pred*= $\lambda P a. a \in Barbs \wedge P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-reachesBarbST*

**by** *simp*

**lemma** (**in** *encoding-wrt-barbs*) *enc-weakly-reflects-barbs-iff-source-target-rel*:

**shows** *enc-weakly-reflects-barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-weakly-reflects-barbs } Rel (STCalWB SWB TWB)$ )

**using** *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**  
*Pred*= $\lambda P a. P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-reachesBarbST*

**by** *simp*

**lemma** (**in** *encoding-wrt-barbs*) *enc-reflects-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**shows** *enc-reflects-barb-set Barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-reflects-barb-set } Rel (STCalWB SWB TWB) Barbs)$

**using** *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**  
*Pred*= $\lambda P a. a \in Barbs \wedge P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

**lemma** (**in** *encoding-wrt-barbs*) *enc-reflects-barbs-iff-source-target-rel*:

**shows** *enc-reflects-barbs*

= ( $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-reflects-barbs } Rel (STCalWB SWB TWB)$ )

**using** *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**  
*Pred*= $\lambda P a. P \Downarrow \langle STCalWB SWB TWB \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

An encoding (weakly) respects barbs iff (1) there exists a relation, like *indRelR*, that relates source

terms and their literal translations and respects (reachability/)existence of barbs, or (2) there exists a relation, like  $\text{indRelL}$ , that relates literal translations and their source terms and respects (reachability/)existence of barbs, or (3) there exists a relation, like  $\text{indRel}$ , that relates source terms and their literal translations in both directions and respects (reachability/)existence of barbs.

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-respects-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**shows** *enc-weakly-respects-barb-set Barbs*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \text{ Barbs})$$

**using** *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**  $\text{Pred} = \lambda P a. a \in \text{Barbs} \wedge P \Downarrow \langle \text{STCalWB SWB TWB} \rangle a$ ] *STCalWB-reachesBarbST*

**by** *simp*

**lemma** (in *encoding-wrt-barbs*) *enc-weakly-respects-barbs-iff-source-target-rel*:

**shows** *enc-weakly-respects-barbs*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-respects-barbs Rel } (\text{STCalWB SWB TWB}))$$

**using** *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**  $\text{Pred} = \lambda P a. P \Downarrow \langle \text{STCalWB SWB TWB} \rangle a$ ] *STCalWB-reachesBarbST*

**by** *simp*

**lemma** (in *encoding-wrt-barbs*) *enc-respects-barb-set-iff-source-target-rel*:

**fixes** *Barbs* :: 'barbs set

**shows** *enc-respects-barb-set Barbs*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \text{ Barbs})$$

**using** *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**  $\text{Pred} = \lambda P a. a \in \text{Barbs} \wedge P \Downarrow \langle \text{STCalWB SWB TWB} \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

**lemma** (in *encoding-wrt-barbs*) *enc-respects-barbs-iff-source-target-rel*:

**shows** *enc-respects-barbs*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-respects-barbs Rel } (\text{STCalWB SWB TWB}))$$

**using** *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**  $\text{Pred} = \lambda P a. P \Downarrow \langle \text{STCalWB SWB TWB} \rangle a$ ] *STCalWB-hasBarbST*

**by** *simp*

Accordingly an encoding is success sensitive iff there exists such a relation between source and target terms that weakly respects the barb success.

**lemma** (in *encoding-wrt-barbs*) *success-sensitive-cond*:

**fixes** *success* :: 'barbs

**shows** *enc-weakly-respects-barb-set {success}* =  $(\forall S. S \Downarrow \langle \text{SWB} \rangle \text{success} \longleftrightarrow \llbracket S \rrbracket \Downarrow \langle \text{TWB} \rangle \text{success})$

**by** *auto*

**lemma** (in *encoding-wrt-barbs*) *success-sensitive-iff-source-target-rel-weakly-respects-success*:

**fixes** *success* :: 'barbs

**shows** *enc-weakly-respects-barb-set {success}*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \{ \text{success} \})$$

**by** (rule *enc-weakly-respects-barb-set-iff-source-target-rel*[**where**  $\text{Barbs} = \{ \text{success} \}$ ])+

**lemma** (in *encoding-wrt-barbs*) *success-sensitive-iff-source-target-rel-respects-success*:

**fixes** *success* :: 'barbs

**shows** *enc-respects-barb-set {success}*

$$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \{ \text{success} \})$$

**by** (rule *enc-respects-barb-set-iff-source-target-rel*[**where**  $\text{Barbs} = \{ \text{success} \}$ ])

**end**

```

theory DivergenceReflection
  imports SourceTargetRelation
begin

```

## 7 Divergence Reflection

Divergence reflection forbids for encodings that introduce loops of internal actions. Thus they determine the practicability of encodings in particular with respect to implementations. An encoding reflects divergence if each loop in a target term result from the translation of a divergent source term.

```

abbreviation (in encoding) enc-preserves-divergence :: bool where
  enc-preserves-divergence  $\equiv$  enc-preserves-pred ( $\lambda P. P \mapsto ST\omega$ )

```

```

lemma (in encoding) divergence-preservation-cond:
  shows enc-preserves-divergence = ( $\forall S. S \mapsto (Source)\omega \longrightarrow \llbracket S \rrbracket \mapsto (Target)\omega$ )
  by simp

```

```

abbreviation (in encoding) enc-reflects-divergence :: bool where
  enc-reflects-divergence  $\equiv$  enc-reflects-pred ( $\lambda P. P \mapsto ST\omega$ )

```

```

lemma (in encoding) divergence-reflection-cond:
  shows enc-reflects-divergence = ( $\forall S. \llbracket S \rrbracket \mapsto (Target)\omega \longrightarrow S \mapsto (Source)\omega$ )
  by simp

```

```

abbreviation rel-preserves-divergence
  :: ('proc  $\times$  'proc) set  $\Rightarrow$  'proc processCalculus  $\Rightarrow$  bool
  where
  rel-preserves-divergence Rel Cal  $\equiv$  rel-preserves-pred Rel ( $\lambda P. P \mapsto (Cal)\omega$ )

```

```

abbreviation rel-reflects-divergence
  :: ('proc  $\times$  'proc) set  $\Rightarrow$  'proc processCalculus  $\Rightarrow$  bool
  where
  rel-reflects-divergence Rel Cal  $\equiv$  rel-reflects-pred Rel ( $\lambda P. P \mapsto (Cal)\omega$ )

```

Apart from divergence reflection we consider divergence respection. An encoding respects divergence if each divergent source term is translated into a divergent target term and each divergent target term result from the translation of a divergent source term.

```

abbreviation (in encoding) enc-respects-divergence :: bool where
  enc-respects-divergence  $\equiv$  enc-respects-pred ( $\lambda P. P \mapsto ST\omega$ )

```

```

lemma (in encoding) divergence-respection-cond:
  shows enc-respects-divergence = ( $\forall S. \llbracket S \rrbracket \mapsto (Target)\omega \longleftrightarrow S \mapsto (Source)\omega$ )
  by auto

```

```

abbreviation rel-respects-divergence
  :: ('proc  $\times$  'proc) set  $\Rightarrow$  'proc processCalculus  $\Rightarrow$  bool
  where
  rel-respects-divergence Rel Cal  $\equiv$  rel-respects-pred Rel ( $\lambda P. P \mapsto (Cal)\omega$ )

```

An encoding preserves divergence iff (1) there exists a relation that relates source terms and their literal translations and preserves divergence, or (2) there exists a relation that relates literal translations and their source terms and reflects divergence.

```

lemma (in encoding) divergence-preservation-iff-source-target-rel-preserves-divergence:
  shows enc-preserves-divergence
    = ( $\exists$  Rel. ( $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in$  Rel)
       $\wedge$  rel-preserves-divergence Rel (STCal Source Target))
  using enc-preserves-pred-iff-source-target-rel-preserves-pred (1)[where Pred= $\lambda P. P \mapsto ST\omega$ ]
  divergentST-STCal-divergent
  by simp

```

**lemma** (in *encoding*) *divergence-preservation-iff-source-target-rel-reflects-divergence*:  
**shows** *enc-preserves-divergence*  
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-reflects-divergence Rel (STCal Source Target))$   
**using** *enc-preserves-pred-iff-source-target-rel-reflects-pred*(1)[**where**  $Pred = \lambda P. P \mapsto ST\omega$ ]  
*divergentST-STCal-divergent*  
**by** *simp*

An encoding reflects divergence iff (1) there exists a relation that relates source terms and their literal translations and reflects divergence, or (2) there exists a relation that relates literal translations and their source terms and preserves divergence.

**lemma** (in *encoding*) *divergence-reflection-iff-source-target-rel-reflects-divergence*:  
**shows** *enc-reflects-divergence*  
 $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-divergence Rel (STCal Source Target))$   
**using** *enc-reflects-pred-iff-source-target-rel-reflects-pred*[**where**  $Pred = \lambda P. P \mapsto ST\omega$ ]  
*divergentST-STCal-divergent*  
**by** *simp*

**lemma** (in *encoding*) *divergence-reflection-iff-source-target-rel-preserves-divergence*:  
**shows** *enc-reflects-divergence*  
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-preserves-divergence Rel (STCal Source Target))$   
**using** *enc-reflects-pred-iff-source-target-rel-preserves-pred*[**where**  $Pred = \lambda P. P \mapsto ST\omega$ ]  
*divergentST-STCal-divergent*  
**by** *simp*

An encoding respects divergence iff there exists a relation that relates source terms and their literal translations in both directions and respects divergence.

**lemma** (in *encoding*) *divergence-respection-iff-source-target-rel-respects-divergence*:  
**shows** *enc-respects-divergence*  $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-divergence Rel (STCal Source Target))$   
**and** *enc-respects-divergence*  $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-respects-divergence Rel (STCal Source Target))$

**proof** –  
**show** *enc-respects-divergence*  $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-divergence Rel (STCal Source Target))$   
**using** *enc-respects-pred-iff-source-target-rel-respects-pred-encR*[**where**  $Pred = \lambda P. P \mapsto ST\omega$ ]  
*divergentST-STCal-divergent*  
**by** *simp*

**next**  
**show** *enc-respects-divergence*  $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel-respects-divergence Rel (STCal Source Target))$   
**using** *enc-respects-pred-iff-source-target-rel-respects-pred-encRL*[**where**  $Pred = \lambda P. P \mapsto ST\omega$ ]  
*divergentST-STCal-divergent*

**by** *simp*

**qed**

**end**

**theory** *OperationalCorrespondence*

**imports** *SourceTargetRelation*

**begin**

## 8 Operational Correspondence

We consider different variants of operational correspondence. This criterion consists of a completeness and a soundness condition and is often defined with respect to a relation  $TRel$  on target terms. Operational completeness modulo  $TRel$  ensures that an encoding preserves source term behaviour modulo  $TRel$  by requiring that each sequence of source term steps can be mimicked by its translation such that the respective derivatives are related by  $TRel$ .

**abbreviation** (in *encoding*) *operational-complete*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*operational-complete*  $TRel \equiv$   
 $\forall S S'. S \mapsto_{Source*} S' \longrightarrow (\exists T. \llbracket S \rrbracket \mapsto_{Target*} T \wedge (\llbracket S \rrbracket, T) \in TRel)$

We call an encoding strongly operational complete modulo  $TRel$  if each source term step has to be mimicked by single target term step of its translation.

**abbreviation** (in *encoding*) *strongly-operational-complete*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*strongly-operational-complete*  $TRel \equiv$   
 $\forall S S'. S \mapsto_{Source} S' \longrightarrow (\exists T. \llbracket S \rrbracket \mapsto_{Target} T \wedge (\llbracket S \rrbracket, T) \in TRel)$

Operational soundness ensures that the encoding does not introduce new behaviour. An encoding is weakly operational sound modulo  $TRel$  if each sequence of target term steps is part of the translation of a sequence of source term steps such that the derivatives are related by  $TRel$ . It allows for intermediate states on the translation of source term step that are not the result of translating a source term.

**abbreviation** (in *encoding*) *weakly-operational-sound*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*weakly-operational-sound*  $TRel \equiv$   
 $\forall S T. \llbracket S \rrbracket \mapsto_{Target*} T \longrightarrow (\exists S' T'. S \mapsto_{Source*} S' \wedge T \mapsto_{Target*} T' \wedge (\llbracket S \rrbracket, T') \in TRel)$

And encoding is operational sound modulo  $TRel$  if each sequence of target term steps is the translation of a sequence of source term steps such that the derivatives are related by  $TRel$ . This criterion does not allow for intermediate states, i.e., does not allow to reach target term from an encoded source term that is not related by  $TRel$  to the translation of a source term.

**abbreviation** (in *encoding*) *operational-sound*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*operational-sound*  $TRel \equiv \forall S T. \llbracket S \rrbracket \mapsto_{Target*} T \longrightarrow (\exists S'. S \mapsto_{Source*} S' \wedge (\llbracket S \rrbracket, T) \in TRel)$

Strong operational soundness modulo  $TRel$  is a stricter variant of operational soundness, where a single target term step has to be mapped on a single source term step.

**abbreviation** (in *encoding*) *strongly-operational-sound*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*strongly-operational-sound*  $TRel \equiv$   
 $\forall S T. \llbracket S \rrbracket \mapsto_{Target} T \longrightarrow (\exists S'. S \mapsto_{Source} S' \wedge (\llbracket S \rrbracket, T) \in TRel)$

An encoding is weakly operational corresponding modulo  $TRel$  if it is operational complete and weakly operational sound modulo  $TRel$ .

**abbreviation** (in *encoding*) *weakly-operational-corresponding*  
 $:: ('procT \times 'procT) set \Rightarrow bool$   
**where**  
*weakly-operational-corresponding*  $TRel \equiv$   
*operational-complete*  $TRel \wedge$  *weakly-operational-sound*  $TRel$

Operational correspondence modulo is the combination of operational completeness and operational soundness modulo  $TRel$ .

**abbreviation** (in *encoding*) *operational-corresponding*  $:: ('procT \times 'procT) set \Rightarrow bool$  **where**  
*operational-corresponding*  $TRel \equiv$  *operational-complete*  $TRel \wedge$  *operational-sound*  $TRel$

An encoding is strongly operational corresponding modulo  $TRel$  if it is strongly operational complete and strongly operational sound modulo  $TRel$ .

**abbreviation** (in *encoding*) *strongly-operational-corresponding*  
 $:: ('procT \times 'procT) set \Rightarrow bool$   
**where**  
*strongly-operational-corresponding*  $TRel \equiv$   
*strongly-operational-complete*  $TRel \wedge$  *strongly-operational-sound*  $TRel$

## 8.1 Trivial Operational Correspondence Results

Every encoding is (weakly) operational corresponding modulo the all relation on target terms.

**lemma** (in *encoding*) *operational-correspondence-modulo-all-relation*:

**shows** *operational-complete*  $\{(T1, T2). \text{True}\}$   
**and** *weakly-operational-sound*  $\{(T1, T2). \text{True}\}$   
**and** *operational-sound*  $\{(T1, T2). \text{True}\}$   
**using** *steps-refl*[**where** *Cal=Source*] *steps-refl*[**where** *Cal=Target*]  
**by** *blast+*

**lemma** *all-relation-is-weak-reduction-bisimulation*:

**fixes** *Cal* :: 'a *processCalculus*  
**shows** *weak-reduction-bisimulation*  $\{(a, b). \text{True}\}$  *Cal*  
**using** *steps-refl*[**where** *Cal=Cal*]  
**by** *blast*

**lemma** (in *encoding*) *operational-correspondence-modulo-some-target-relation*:

**shows**  $\exists TRel. \text{weakly-operational-corresponding } TRel$   
**and**  $\exists TRel. \text{operational-corresponding } TRel$   
**and**  $\exists TRel. \text{weakly-operational-corresponding } TRel \wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$   
**and**  $\exists TRel. \text{operational-corresponding } TRel \wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$   
**using** *operational-correspondence-modulo-all-relation*  
*all-relation-is-weak-reduction-bisimulation*[**where** *Cal=Target*]  
**by** *blast+*

Strong operational correspondence requires that source can perform a step iff their translations can perform a step.

**lemma** (in *encoding*) *strong-operational-correspondence-modulo-some-target-relation*:

**shows**  $(\exists TRel. \text{strongly-operational-corresponding } TRel)$   
 $= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$   
**and**  $(\exists TRel. \text{strongly-operational-corresponding } TRel$   
 $\wedge \text{weak-reduction-bisimulation } TRel \text{ Target})$   
 $= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$

**proof** –

**have** *A1*:  $\exists TRel. \text{strongly-operational-corresponding } TRel$   
 $\implies \forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T)$   
**by** *blast*

**moreover have** *A2*:  $\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T)$   
 $\implies \exists TRel. \text{strongly-operational-corresponding } TRel$   
 $\wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$

**proof** –

**assume**  $\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T)$

**hence** *strongly-operational-corresponding*  $\{(T1, T2). \text{True}\}$

**by** *simp*

**thus**  $\exists TRel. \text{strongly-operational-corresponding } TRel$

$\wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$

**using** *all-relation-is-weak-reduction-bisimulation*[**where** *Cal=Target*]

**by** *blast*

**qed**

**ultimately show**  $(\exists TRel. \text{strongly-operational-corresponding } TRel$   
 $\wedge \text{weak-reduction-bisimulation } TRel \text{ Target})$   
 $= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$

**by** *blast*

**from** *A1 A2* **show**  $(\exists TRel. \text{strongly-operational-corresponding } TRel)$

$= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$

**by** *blast*

**qed**

## 8.2 (Strong) Operational Completeness vs (Strong) Simulation

An encoding is operational complete modulo a weak simulation on target terms  $TRel$  iff there is a relation, like  $indRelRTPO$ , that relates at least all source terms to their literal translations, includes  $TRel$ , and is a weak simulation.

**lemma** (in *encoding*) *weak-reduction-simulation-impl-OCOM*:  
**fixes**  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc$  set  
**and**  $TRel :: ('procT \times 'procT)$  set  
**assumes**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $A3: weak-reduction-simulation\ Rel\ (STCal\ Source\ Target)$   
**shows** *operational-complete* ( $TRel^*$ )  
**proof** *clarify*  
**fix**  $S\ S'$   
**from**  $A1$  **have**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**by** *simp*  
**moreover** **assume**  $S \mapsto Source^* S'$   
**hence**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)^* (SourceTerm\ S')$   
**by** (*simp add: STCal-steps(1)*)  
**ultimately** **obtain**  $Q'$  **where**  $A5: TargetTerm\ (\llbracket S \rrbracket) \mapsto (STCal\ Source\ Target)^* Q'$   
**and**  $A6: (SourceTerm\ S', Q') \in Rel$   
**using**  $A3$   
**by** *blast*  
**from**  $A5$  **obtain**  $T$  **where**  $A7: T \in T\ Q'$  **and**  $A8: \llbracket S \rrbracket \mapsto Target^* T$   
**by** (*auto simp add: STCal-steps(2)*)  
**from**  $A2\ A6\ A7$  **have**  $(\llbracket S \rrbracket, T) \in TRel^*$   
**by** *simp*  
**with**  $A8$  **show**  $\exists T. \llbracket S \rrbracket \mapsto Target^* T \wedge (\llbracket S \rrbracket, T) \in TRel^*$   
**by** *blast*  
**qed**

**lemma** (in *encoding*) *OCOM-iff-indRelRTPO-is-weak-reduction-simulation*:  
**fixes**  $TRel :: ('procT \times 'procT)$  set  
**shows** (*operational-complete* ( $TRel^*$ )  
 $\wedge$  *weak-reduction-simulation* ( $TRel^+$ )  $Target$ )  
 $=$  *weak-reduction-simulation* ( $indRelRTPO\ TRel$ ) ( $STCal\ Source\ Target$ )  
**proof** (*rule iffI, erule conjE*)  
**assume**  $oc: operational-complete\ (TRel^*)$   
**and**  $sim: weak-reduction-simulation\ (TRel^+) Target$   
**show** *weak-reduction-simulation* ( $indRelRTPO\ TRel$ ) ( $STCal\ Source\ Target$ )  
**proof** *clarify*  
**fix**  $P\ Q\ P'$   
**assume**  $P \lesssim [\cdot]RT < TRel > Q$  **and**  $P \mapsto (STCal\ Source\ Target)^* P'$   
**thus**  $\exists Q'. Q \mapsto (STCal\ Source\ Target)^* Q' \wedge P' \lesssim [\cdot]RT < TRel > Q'$   
**proof** (*induct arbitrary: P'*)  
**case** (*encR S*)  
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)^* P'$   
**from** *this* **obtain**  $S'$  **where**  $A1: S' \in S\ P'$  **and**  $A2: S \mapsto Source^* S'$   
**by** (*auto simp add: STCal-steps(1)*)  
**from**  $oc\ A2$  **obtain**  $T$  **where**  $A3: \llbracket S \rrbracket \mapsto Target^* T$  **and**  $A4: (\llbracket S \rrbracket, T) \in TRel^*$   
**by** *blast*  
**from**  $A3$  **have**  $TargetTerm\ (\llbracket S \rrbracket) \mapsto (STCal\ Source\ Target)^* (TargetTerm\ T)$   
**by** (*simp add: STCal-steps(2)*)  
**moreover** **have**  $P' \lesssim [\cdot]RT < TRel > TargetTerm\ T$   
**proof** –  
**from**  $A4$  **have**  $\llbracket S \rrbracket = T \vee (\llbracket S \rrbracket, T) \in TRel^+$   
**using** *rtrancl-eq-or-trancl*[of  $\llbracket S \rrbracket\ T\ TRel$ ]  
**by** *blast*  
**moreover** **from**  $A1$  **have**  $A5: P' \lesssim [\cdot]RT < TRel > TargetTerm\ (\llbracket S \rrbracket)$   
**by** (*simp add: indRelRTPO.encR*)

hence  $\llbracket S \rrbracket = T \implies P' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
 by *simp*  
 moreover have  $(\llbracket S \rrbracket, T) \in TRel^+ \implies P' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
**proof** –  
 assume  $(\llbracket S \rrbracket, T) \in TRel^+$   
 hence  $TargetTerm (\llbracket S \rrbracket) \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
**proof** *induct*  
 fix  $T$   
 assume  $(\llbracket S \rrbracket, T) \in TRel$   
 thus  $TargetTerm (\llbracket S \rrbracket) \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
 by (rule *indRelRTPO.target*)  
**next**  
 case (step  $TQ TR$ )  
 assume  $TargetTerm (\llbracket S \rrbracket) \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm TQ$   
 moreover assume  $(TQ, TR) \in TRel$   
 hence  $TargetTerm TQ \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm TR$   
 by (rule *indRelRTPO.target*)  
 ultimately show  $TargetTerm (\llbracket S \rrbracket) \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm TR$   
 by (rule *indRelRTPO.trans*)  
**qed**  
 with  $A5$  show  $P' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
 by (rule *indRelRTPO.trans*)  
**qed**  
 ultimately show  $P' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T$   
 by *blast*  
**qed**  
 ultimately  
 show  $\exists Q'. TargetTerm (\llbracket S \rrbracket) \mapsto (STCal Source Target)^* Q' \wedge P' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q'$   
 by *blast*  
**next**  
 case (source  $S$ )  
 then obtain  $S'$  where  $B1: S' \in S P'$   
 by (auto *simp add: STCal-steps(1)*)  
 hence  $P' \lesssim \llbracket \cdot \rrbracket RT < TRel > P'$   
 by (*simp add: indRelRTPO.source*)  
 with source show  $\exists Q'. SourceTerm S \mapsto (STCal Source Target)^* Q' \wedge P' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q'$   
 by *blast*  
**next**  
 case (target  $T1 T2$ )  
 assume  $TargetTerm T1 \mapsto (STCal Source Target)^* P'$   
 from this obtain  $T1'$  where  $C1: T1' \in T P'$  and  $C2: T1 \mapsto Target^* T1'$   
 by (auto *simp add: STCal-steps(2)*)  
 assume  $(T1, T2) \in TRel$   
 hence  $(T1, T2) \in TRel^+$   
 by *simp*  
 with  $C2$  *sim* obtain  $T2'$  where  $C3: T2 \mapsto Target^* T2'$   
 and  $C4: (T1', T2') \in TRel^+$   
 by *blast*  
 from  $C3$  have  $TargetTerm T2 \mapsto (STCal Source Target)^* (TargetTerm T2')$   
 by (*simp add: STCal-steps(2)*)  
 moreover from  $C4$  have  $TargetTerm T1' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T2'$   
**proof** *induct*  
 fix  $T2'$   
 assume  $(T1', T2') \in TRel$   
 thus  $TargetTerm T1' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T2'$   
 by (rule *indRelRTPO.target*)  
**next**  
 case (step  $TQ TR$ )  
 assume  $TargetTerm T1' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm TQ$   
 moreover assume  $(TQ, TR) \in TRel$   
 hence  $TargetTerm TQ \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm TR$



by (rule indRelRTPO.target)  
 ultimately show TargetTerm T1'  $\lesssim_{[\cdot]} RT < TRel >$  TargetTerm TR  
 by (rule indRelRTPO.trans)  
 qed  
 with C1 have P'  $\lesssim_{[\cdot]} RT < TRel >$  TargetTerm T2'  
 by simp  
 ultimately show  $\exists Q'. \text{TargetTerm } T2 \mapsto (\text{STCal Source Target})^* Q' \wedge P' \lesssim_{[\cdot]} RT < TRel > Q'$   
 by blast  
 next  
 case (trans P Q R)  
 assume P  $\mapsto (\text{STCal Source Target})^* P'$   
 and  $\bigwedge P'. P \mapsto (\text{STCal Source Target})^* P'$   
 $\implies \exists Q'. Q \mapsto (\text{STCal Source Target})^* Q' \wedge P' \lesssim_{[\cdot]} RT < TRel > Q'$   
 from this obtain Q' where D1: Q  $\mapsto (\text{STCal Source Target})^* Q'$   
 and D2: P'  $\lesssim_{[\cdot]} RT < TRel > Q'$   
 by blast  
 assume  $\bigwedge Q'. Q \mapsto (\text{STCal Source Target})^* Q'$   
 $\implies \exists R'. R \mapsto (\text{STCal Source Target})^* R' \wedge Q' \lesssim_{[\cdot]} RT < TRel > R'$   
 with D1 obtain R' where D3: R  $\mapsto (\text{STCal Source Target})^* R'$   
 and D4: Q'  $\lesssim_{[\cdot]} RT < TRel > R'$   
 by blast  
 from D2 D4 have P'  $\lesssim_{[\cdot]} RT < TRel > R'$   
 by (rule indRelRTPO.trans)  
 with D3 show  $\exists R'. R \mapsto (\text{STCal Source Target})^* R' \wedge P' \lesssim_{[\cdot]} RT < TRel > R'$   
 by blast  
 qed  
 qed  
 next  
 have  $\forall S. \text{SourceTerm } S \lesssim_{[\cdot]} RT < TRel > \text{TargetTerm } (\llbracket S \rrbracket)$   
 by (simp add: indRelRTPO.encR)  
 moreover have  $\forall S T. \text{SourceTerm } S \lesssim_{[\cdot]} RT < TRel > \text{TargetTerm } T \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
 using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond  
 by simp  
 moreover assume sim: weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)  
 ultimately have operational-complete (TRel\*)  
 using weak-reduction-simulation-impl-OCOM[where Rel=indRelRTPO TRel and TRel=TRel]  
 by simp  
 moreover from sim have weak-reduction-simulation (TRel<sup>+</sup>) Target  
 using indRelRTPO-impl-TRel-is-weak-reduction-simulation[where TRel=TRel]  
 by simp  
 ultimately show operational-complete (TRel\*)  
 $\wedge$  weak-reduction-simulation (TRel<sup>+</sup>) Target  
 by simp  
 qed  
 lemma (in encoding) OCom-iff-weak-reduction-simulation:  
 fixes TRel :: ('procT  $\times$  'procT) set  
 shows (operational-complete (TRel\*))  
 $\wedge$  weak-reduction-simulation (TRel<sup>+</sup>) Target  
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-simulation } \text{Rel } (\text{STCal Source Target}))$   
 proof (rule iffI, erule conjE)  
 have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } \text{TRel}$   
 by (simp add: indRelRTPO.encR)  
 moreover have  $\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow \text{TargetTerm } T1 \lesssim_{[\cdot]} RT < TRel > \text{TargetTerm } T2$   
 by (simp add: indRelRTPO.target)  
 moreover have  $\forall T1 T2. \text{TargetTerm } T1 \lesssim_{[\cdot]} RT < TRel > \text{TargetTerm } T2 \longrightarrow (T1, T2) \in \text{TRel}^+$   
 using indRelRTPO-to-TRel(4)[where TRel=TRel]

**by** *simp*  
**moreover have**  $\forall S T. \text{SourceTerm } S \lesssim [\![S]\!] \text{RT} < \text{TRel} > \text{TargetTerm } T \longrightarrow ([\![S]\!], T) \in \text{TRel}^*$   
**using** *indRelRTPO-to-TRel(2)[where TRel=TRel]* *trans-closure-of-TRel-refl-cond*  
**by** *simp*  
**moreover assume** *operational-complete (TRel\*)*  
**and** *weak-reduction-simulation (TRel+) Target*  
**hence** *weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)*  
**using** *OCom-iff-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]*  
**by** *simp*  
**ultimately show**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!])) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([\![S]\!], T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-simulation } \text{Rel} \text{ (STCal Source Target)}$   
**by** *blast*  
**next**  
**assume**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!])) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([\![S]\!], T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-simulation } \text{Rel} \text{ (STCal Source Target)}$   
**from this obtain** *Rel* **where**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!])) \in \text{Rel}$   
**and**  $A2: \forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$   
**and**  $A3: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+$   
**and**  $A4: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([\![S]\!], T) \in \text{TRel}^*$   
**and**  $A5: \text{weak-reduction-simulation } \text{Rel} \text{ (STCal Source Target)}$   
**by** *blast*  
**from**  $A1 A4 A5$  **have** *operational-complete (TRel\*)*  
**using** *weak-reduction-simulation-impl-OCom[where Rel=Rel and TRel=TRel]*  
**by** *simp*  
**moreover from**  $A2 A3 A5$  **have** *weak-reduction-simulation (TRel+) Target*  
**using** *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation[where Rel=Rel and TRel=TRel]*  
**by** *simp*  
**ultimately show** *operational-complete (TRel\*)*  
 $\wedge \text{weak-reduction-simulation (TRel+) Target}$   
**by** *simp*  
**qed**

An encoding is strong operational complete modulo a strong simulation on target terms TRel iff there is a relation, like indRelRTPO, that relates at least all source terms to their literal translations, includes TRel, and is a strong simulation.

**lemma** (*in encoding*) *strong-reduction-simulation-impl-SOCom:*

**fixes**  $\text{Rel} :: (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{ set}$

**and**  $\text{TRel} :: ('procT \times 'procT) \text{ set}$

**assumes**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!])) \in \text{Rel}$

**and**  $A2: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([\![S]\!], T) \in \text{TRel}^*$

**and**  $A3: \text{strong-reduction-simulation } \text{Rel} \text{ (STCal Source Target)}$

**shows** *strongly-operational-complete (TRel\*)*

**proof** *clarify*

**fix**  $S S'$

**from**  $A1$  **have**  $(\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!])) \in \text{Rel}$

**by** *simp*

**moreover assume**  $S \mapsto \text{Source } S'$

**hence**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target}) (\text{SourceTerm } S')$

**by** (*simp add: STCal-step(1)*)

**ultimately obtain**  $Q'$  **where**  $A5: \text{TargetTerm } ([\![S]\!]) \mapsto (\text{STCal Source Target}) Q'$

**and**  $A6: (\text{SourceTerm } S', Q') \in \text{Rel}$

**using**  $A3$

**by** *blast*

**from**  $A5$  **obtain**  $T$  **where**  $A7: T \in T$   $Q'$  **and**  $A8: \llbracket S \rrbracket \mapsto \text{Target } T$   
**by** (*auto simp add: STCal-step(2)*)  
**from**  $A2$   $A6$   $A7$  **have**  $(\llbracket S \rrbracket, T) \in TRel^*$   
**by** *simp*  
**with**  $A8$  **show**  $\exists T. \llbracket S \rrbracket \mapsto \text{Target } T \wedge (\llbracket S \rrbracket, T) \in TRel^*$   
**by** *blast*  
**qed**

**lemma** (*in encoding*) *SOCOM-iff-indRelRTPO-is-strong-reduction-simulation:*

**fixes**  $TRel :: ('procT \times 'procT)$  *set*

**shows** (*strongly-operational-complete*  $(TRel^*)$ )

$\wedge$  *strong-reduction-simulation*  $(TRel^+) \text{ Target}$

$=$  *strong-reduction-simulation*  $(\text{indRelRTPO } TRel) (\text{STCal Source Target})$

**proof** (*rule iffI, erule conjE*)

**assume** *soc: strongly-operational-complete*  $(TRel^*)$

**and** *sim: strong-reduction-simulation*  $(TRel^+) \text{ Target}$

**show** *strong-reduction-simulation*  $(\text{indRelRTPO } TRel) (\text{STCal Source Target})$

**proof** *clarify*

**fix**  $P$   $Q$   $P'$

**assume**  $P \lesssim [\cdot]RT < TRel > Q$  **and**  $P \mapsto (\text{STCal Source Target}) P'$

**thus**  $\exists Q'. Q \mapsto (\text{STCal Source Target}) Q' \wedge P' \lesssim [\cdot]RT < TRel > Q'$

**proof** (*induct arbitrary: P'*)

**case** (*encR S*)

**assume** *SourceTerm S*  $\mapsto (\text{STCal Source Target}) P'$

**from this obtain**  $S'$  **where**  $A1: S' \in S$   $P'$  **and**  $A2: S \mapsto \text{Source } S'$

**by** (*auto simp add: STCal-step(1)*)

**from** *soc A2 obtain T where*  $A3: \llbracket S \rrbracket \mapsto \text{Target } T$  **and**  $A4: (\llbracket S \rrbracket, T) \in TRel^*$

**by** *blast*

**from**  $A3$  **have** *TargetTerm*  $(\llbracket S \rrbracket) \mapsto (\text{STCal Source Target}) (\text{TargetTerm } T)$

**by** (*simp add: STCal-step(2)*)

**moreover have**  $P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

**proof** –

**from**  $A4$  **have**  $\llbracket S \rrbracket = T \vee (\llbracket S \rrbracket, T) \in TRel^+$

**using** *rtrancl-eq-or-trancl*[*of*  $\llbracket S \rrbracket$   $T$   $TRel$ ]

**by** *blast*

**moreover from**  $A1$  **have**  $A5: P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } (\llbracket S \rrbracket)$

**by** (*simp add: indRelRTPO.encR*)

**hence**  $\llbracket S \rrbracket = T \implies P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

**by** *simp*

**moreover have**  $(\llbracket S \rrbracket, T) \in TRel^+ \implies P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

**proof** –

**assume**  $(\llbracket S \rrbracket, T) \in TRel^+$

**hence** *TargetTerm*  $(\llbracket S \rrbracket) \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

**proof** *induct*

**fix**  $TQ$

**assume**  $(\llbracket S \rrbracket, TQ) \in TRel$

**thus** *TargetTerm*  $(\llbracket S \rrbracket) \lesssim [\cdot]RT < TRel > \text{TargetTerm } TQ$

**by** (*rule indRelRTPO.target*)

**next**

**case** (*step TQ TR*)

**assume** *TargetTerm*  $(\llbracket S \rrbracket) \lesssim [\cdot]RT < TRel > \text{TargetTerm } TQ$

**moreover assume**  $(TQ, TR) \in TRel$

**hence** *TargetTerm*  $TQ \lesssim [\cdot]RT < TRel > \text{TargetTerm } TR$

**by** (*rule indRelRTPO.target*)

**ultimately show** *TargetTerm*  $(\llbracket S \rrbracket) \lesssim [\cdot]RT < TRel > \text{TargetTerm } TR$

**by** (*rule indRelRTPO.trans*)

**qed**

**with**  $A5$  **show**  $P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

**by** (*rule indRelRTPO.trans*)

**qed**

**ultimately show**  $P' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T$

by *blast*  
**qed**  
**ultimately**  
**show**  $\exists Q'. \text{TargetTerm } (\llbracket S \rrbracket) \mapsto (\text{STCal Source Target}) Q' \wedge P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > Q'$   
 by *blast*  
**next**  
 case (*source S*)  
**then obtain**  $S'$  **where**  $B1: S' \in S P'$   
 by (*auto simp add: STCal-step(1)*)  
**hence**  $P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > P'$   
 by (*simp add: indRelRTPO.source*)  
**with source show**  $\exists Q'. \text{SourceTerm } S \mapsto (\text{STCal Source Target}) Q' \wedge P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > Q'$   
 by *blast*  
**next**  
 case (*target T1 T2*)  
**assume**  $\text{TargetTerm } T1 \mapsto (\text{STCal Source Target}) P'$   
**from this obtain**  $T1'$  **where**  $C1: T1' \in T P'$  **and**  $C2: T1 \mapsto \text{Target } T1'$   
 by (*auto simp add: STCal-step(2)*)  
**assume**  $(T1, T2) \in \text{TRel}$   
**hence**  $(T1, T2) \in \text{TRel}^+$   
 by *simp*  
**with C2 sim obtain**  $T2'$  **where**  $C3: T2 \mapsto \text{Target } T2'$  **and**  $C4: (T1', T2') \in \text{TRel}^+$   
 by *blast*  
**from C3 have**  $\text{TargetTerm } T2 \mapsto (\text{STCal Source Target}) (\text{TargetTerm } T2')$   
 by (*simp add: STCal-step(2)*)  
**moreover from C4 have**  $\text{TargetTerm } T1' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } T2'$   
**proof induct**  
 fix  $T2'$   
**assume**  $(T1', T2') \in \text{TRel}$   
**thus**  $\text{TargetTerm } T1' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } T2'$   
 by (*rule indRelRTPO.target*)  
**next**  
 case (*step TQ TR*)  
**assume**  $\text{TargetTerm } T1' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } TQ$   
**moreover assume**  $(TQ, TR) \in \text{TRel}$   
**hence**  $\text{TargetTerm } TQ \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } TR$   
 by (*rule indRelRTPO.target*)  
**ultimately show**  $\text{TargetTerm } T1' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } TR$   
 by (*rule indRelRTPO.trans*)  
**qed**  
**with C1 have**  $P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > \text{TargetTerm } T2'$   
 by *simp*  
**ultimately show**  $\exists Q'. \text{TargetTerm } T2 \mapsto (\text{STCal Source Target}) Q' \wedge P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > Q'$   
 by *blast*  
**next**  
 case (*trans P Q R*)  
**assume**  $P \mapsto (\text{STCal Source Target}) P'$   
**and**  $\bigwedge P'. P \mapsto (\text{STCal Source Target}) P'$   
 $\implies \exists Q'. Q \mapsto (\text{STCal Source Target}) Q' \wedge P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > Q'$   
**from this obtain**  $Q'$  **where**  $D1: Q \mapsto (\text{STCal Source Target}) Q'$   
**and**  $D2: P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > Q'$   
 by *blast*  
**assume**  $\bigwedge Q'. Q \mapsto (\text{STCal Source Target}) Q'$   
 $\implies \exists R'. R \mapsto (\text{STCal Source Target}) R' \wedge Q' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > R'$   
**with D1 obtain**  $R'$  **where**  $D3: R \mapsto (\text{STCal Source Target}) R'$   
**and**  $D4: Q' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > R'$   
 by *blast*  
**from D2 D4 have**  $P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > R'$   
 by (*rule indRelRTPO.trans*)  
**with D3 show**  $\exists R'. R \mapsto (\text{STCal Source Target}) R' \wedge P' \lesssim \llbracket \cdot \rrbracket \text{RT} < \text{TRel} > R'$   
 by *blast*

**qed**  
**qed**  
**next**  
**have**  $\forall S. \text{SourceTerm } S \lesssim[\cdot]RT\langle TRel \rangle \text{TargetTerm } (\llbracket S \rrbracket)$   
**by** (*simp add: indRelRTPO.encR*)  
**moreover have**  $\forall S T. \text{SourceTerm } S \lesssim[\cdot]RT\langle TRel \rangle \text{TargetTerm } T \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**using** *indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond*  
**by** *simp*  
**moreover assume** *sim: strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)*  
**ultimately have** *strongly-operational-complete (TRel\*)*  
**using** *strong-reduction-simulation-impl-SOCom[where Rel=indRelRTPO TRel and TRel=TRel]*  
**by** *simp*  
**moreover from** *sim have strong-reduction-simulation (TRel+) Target*  
**using** *indRelRTPO-impl-TRel-is-strong-reduction-simulation[where TRel=TRel]*  
**by** *simp*  
**ultimately show** *strongly-operational-complete (TRel\*)*  
 $\wedge$  *strong-reduction-simulation (TRel+) Target*  
**by** *simp*  
**qed**

**lemma** (*in encoding*) *SOCom-iff-strong-reduction-simulation:*

**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$

**shows** (*strongly-operational-complete (TRel\*)*

$\wedge$  *strong-reduction-simulation (TRel+) Target*

$= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$

$\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$

$\wedge$  *strong-reduction-simulation Rel (STCal Source Target)*)

**proof** (*rule iffI, erule conjE*)

**have**  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } TRel$

**by** (*simp add: indRelRTPO.encR*)

**moreover have**  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow \text{TargetTerm } T1 \lesssim[\cdot]RT\langle TRel \rangle \text{TargetTerm } T2$

**by** (*simp add: indRelRTPO.target*)

**moreover have**  $\forall T1 T2. \text{TargetTerm } T1 \lesssim[\cdot]RT\langle TRel \rangle \text{TargetTerm } T2 \longrightarrow (T1, T2) \in TRel^+$

**using** *indRelRTPO-to-TRel(4)[where TRel=TRel]*

**by** *simp*

**moreover have**  $\forall S T. \text{SourceTerm } S \lesssim[\cdot]RT\langle TRel \rangle \text{TargetTerm } T \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$

**using** *indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond*

**by** *simp*

**moreover assume** *strongly-operational-complete (TRel\*)*

**and** *strong-reduction-simulation (TRel+) Target*

**hence** *strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)*

**using** *SOCom-iff-indRelRTPO-is-strong-reduction-simulation[where TRel=TRel]*

**by** *simp*

**ultimately show**  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$

$\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$

$\wedge$  *strong-reduction-simulation Rel (STCal Source Target)*

**by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$

$\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$

$\wedge$  *strong-reduction-simulation Rel (STCal Source Target)*

**from this obtain** *Rel* **where**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: \forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel$

**and**  $A3: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$

**and**  $A4: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$

**and**  $A5$ : *strong-reduction-simulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**by** *blast*  
**from**  $A1$   $A4$   $A5$  **have** *strongly-operational-complete* ( $TRel^*$ )  
**using** *strong-reduction-simulation-impl-SOCom*[**where**  $Rel=Rel$  **and**  $TRel=TRel$ ]  
**by** *simp*  
**moreover from**  $A2$   $A3$   $A5$  **have** *strong-reduction-simulation* ( $TRel^+$ )  $Target$   
**using** *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation*[**where**  $Rel=Rel$  **and**  
 $TRel=TRel$ ]  
**by** *simp*  
**ultimately show** *strongly-operational-complete* ( $TRel^*$ )  
 $\wedge$  *strong-reduction-simulation* ( $TRel^+$ )  $Target$   
**by** *simp*  
**qed**

**lemma** (**in** *encoding*) *target-relation-from-source-target-relation*:  
**fixes**  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$   
**assumes**  $stre: \forall S T. (SourceTerm S, TargetTerm T) \in Rel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=$   
**shows**  $\exists TRel. (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel)$   
 $\wedge (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$

**proof** –  
**define**  $TRel$  **where**  $TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$   
**from**  $TRel-def$  **have**  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**by** *simp*  
**moreover from**  $TRel-def$   
**have**  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**by** *blast*  
**moreover from**  $stre$   $TRel-def$   
**have**  $\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**by** *blast*  
**ultimately show** *?thesis*  
**by** *blast*  
**qed**

**lemma** (**in** *encoding*) *SOCom-modulo-TRel-iff-strong-reduction-simulation*:  
**shows**  $(\exists TRel. \text{strongly-operational-complete } (TRel^*))$   
 $\wedge$  *strong-reduction-simulation* ( $TRel^+$ )  $Target$   
 $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=)$   
 $\wedge \text{strong-reduction-simulation } Rel (STCal Source Target))$

**proof** (*rule iffI*)  
**assume**  $\exists TRel. \text{strongly-operational-complete } (TRel^*)$   
 $\wedge$  *strong-reduction-simulation* ( $TRel^+$ )  $Target$   
**from** *this* **obtain**  $TRel$  **where** *strongly-operational-complete* ( $TRel^*$ )  
**and** *strong-reduction-simulation* ( $TRel^+$ )  $Target$   
**by** *blast*  
**hence** *strong-reduction-simulation* ( $indRelRTPO$   $TRel$ ) ( $STCal$   $Source$   $Target$ )  
**using** *SOCom-iff-indRelRTPO-is-strong-reduction-simulation*[**where**  $TRel=TRel$ ]  
**by** *simp*  
**moreover have**  $\forall S. SourceTerm S \lesssim [\cdot]_{RT} < TRel > TargetTerm (\llbracket S \rrbracket)$   
**by** (*simp add: indRelRTPO.encR*)  
**moreover have**  $\forall S T. SourceTerm S \lesssim [\cdot]_{RT} < TRel > TargetTerm T$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in (indRelRTPO TRel)^=$   
**using** *indRelRTPO-relates-source-target*[**where**  $TRel=TRel$ ]  
**by** *simp*  
**ultimately show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=)$   
 $\wedge$  *strong-reduction-simulation*  $Rel (STCal Source Target)$   
**by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (TargetTerm\ (\llbracket S \rrbracket), TargetTerm\ T) \in Rel^=)$   
 $\wedge strong\text{-reduction-simulation}\ Rel\ (STCal\ Source\ Target)$   
**from this obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel$   
 $\longrightarrow (TargetTerm\ (\llbracket S \rrbracket), TargetTerm\ T) \in Rel^=)$   
**and**  $A3: strong\text{-reduction-simulation}\ Rel\ (STCal\ Source\ Target)$   
**by** *blast*  
**from**  $A2$  **obtain**  $TRel$  **where**  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**using** *target-relation-from-source-target-relation*[**where**  $Rel=Rel$ ]  
**by** *blast*  
**with**  $A1\ A3$  **have** *strongly-operational-complete*  $(TRel^*)$   
 $\wedge strong\text{-reduction-simulation}\ (TRel^+)\ Target$   
**using** *SOCom-iff-strong-reduction-simulation*[**where**  $TRel=TRel$ ]  
**by** *blast*  
**thus**  $\exists TRel. strong\text{-operational-complete}\ (TRel^*)$   
 $\wedge strong\text{-reduction-simulation}\ (TRel^+)\ Target$   
**by** *blast*  
**qed**

### 8.3 Weak Operational Soundness vs Contrasimulation

If the inverse of a relation that includes  $TRel$  and relates source terms and their literal translations is a contrasimulation, then the encoding is weakly operational sound.

**lemma** (in *encoding*) *weak-reduction-contrasimulation-impl-WOSou*:

**fixes**  $Rel :: ('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc$  *set*  
**and**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $A3: weak\text{-reduction-contrasimulation}\ (Rel^{-1})\ (STCal\ Source\ Target)$   
**shows** *weakly-operational-sound*  $(TRel^*)$   
**proof** *clarify*  
**fix**  $S\ T$   
**from**  $A1$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel^{-1}$   
**by** *simp*  
**moreover** **assume**  $\llbracket S \rrbracket \mapsto Target^*\ T$   
**hence**  $TargetTerm\ (\llbracket S \rrbracket) \mapsto (STCal\ Source\ Target)^*\ (TargetTerm\ T)$   
**by** (*simp add: STCal-steps(2)*)  
**ultimately obtain**  $Q'$  **where**  $A5: SourceTerm\ S \mapsto (STCal\ Source\ Target)^*\ Q'$   
**and**  $A6: (Q', TargetTerm\ T) \in Rel^{-1}$   
**using**  $A3$   
**by** *blast*  
**from**  $A5$  **obtain**  $S'$  **where**  $A7: S' \in S\ Q'$  **and**  $A8: S \mapsto Source^*\ S'$   
**by** (*auto simp add: STCal-steps(1)*)  
**have**  $Q' \mapsto (STCal\ Source\ Target)^*\ Q'$   
**by** (*simp add: steps-refl*)  
**with**  $A6\ A3$  **obtain**  $P''$  **where**  $A9: TargetTerm\ T \mapsto (STCal\ Source\ Target)^*\ P''$   
**and**  $A10: (P'', Q') \in Rel^{-1}$   
**by** *blast*  
**from**  $A9$  **obtain**  $T'$  **where**  $A11: T' \in T\ P''$  **and**  $A12: T \mapsto Target^*\ T'$   
**by** (*auto simp add: STCal-steps(2)*)  
**from**  $A10$  **have**  $(Q', P'') \in Rel$   
**by** *induct*  
**with**  $A2\ A7\ A11$  **have**  $(\llbracket S \rrbracket, T') \in TRel^*$   
**by** *simp*  
**with**  $A8\ A12$  **show**  $\exists S'\ T'. S \mapsto Source^*\ S' \wedge T \mapsto Target^*\ T' \wedge (\llbracket S \rrbracket, T') \in TRel^*$   
**by** *blast*

qed

## 8.4 (Strong) Operational Soundness vs (Strong) Simulation

An encoding is operational sound modulo a relation  $TRel$  whose inverse is a weak reduction simulation on target terms iff there is a relation, like  $indRelRTPO$ , that relates at least all source terms to their literal translations, includes  $TRel$ , and whose inverse is a weak simulation.

**lemma** (in encoding) *weak-reduction-simulation-impl-OSou*:

**fixes**  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc$  set

**and**  $TRel :: ('procT \times 'procT)$  set

**assumes**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

**and**  $A2: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$

**and**  $A3: \text{weak-reduction-simulation } (Rel^{-1}) (STCal\ Source\ Target)$

**shows** *operational-sound* ( $TRel^*$ )

**proof** *clarify*

**fix**  $S\ T$

**from**  $A1$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel^{-1}$

**by** *simp*

**moreover** **assume**  $\llbracket S \rrbracket \longmapsto Target^*\ T$

**hence**  $TargetTerm\ (\llbracket S \rrbracket) \longmapsto (STCal\ Source\ Target)^*\ (TargetTerm\ T)$

**by** (*simp add: STCal-steps(2)*)

**ultimately** **obtain**  $Q'$  **where**  $A5: SourceTerm\ S \longmapsto (STCal\ Source\ Target)^*\ Q'$

**and**  $A6: (TargetTerm\ T, Q') \in Rel^{-1}$

**using**  $A3$

**by** *blast*

**from**  $A5$  **obtain**  $S'$  **where**  $A7: S' \in S\ Q'$  **and**  $A8: S \longmapsto Source^*\ S'$

**by** (*auto simp add: STCal-steps(1)*)

**from**  $A6$  **have**  $(Q', TargetTerm\ T) \in Rel$

**by** *induct*

**with**  $A2\ A7$  **have**  $(\llbracket S \rrbracket, T) \in TRel^*$

**by** *simp*

**with**  $A8$  **show**  $\exists S'. S \longmapsto Source^*\ S' \wedge (\llbracket S \rrbracket, T) \in TRel^*$

**by** *blast*

qed

**lemma** (in encoding) *OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation*:

**fixes**  $TRel :: ('procT \times 'procT)$  set

**shows** (*operational-sound* ( $TRel^*$ ))

$\wedge$  *weak-reduction-simulation*  $((TRel^+)^{-1})\ Target$

$=$  *weak-reduction-simulation*  $((indRelRTPO\ TRel)^{-1})\ (STCal\ Source\ Target)$

**proof** (*rule iffI, erule conjE*)

**assume** *os*: *operational-sound* ( $TRel^*$ )

**and** *sim*: *weak-reduction-simulation*  $((TRel^+)^{-1})\ Target$

**show** *weak-reduction-simulation*  $((indRelRTPO\ TRel)^{-1})\ (STCal\ Source\ Target)$

**proof** *clarify*

**fix**  $P\ Q\ P'$

**assume**  $Q \lesssim [\cdot]RT < TRel > P$  **and**  $P \longmapsto (STCal\ Source\ Target)^*\ P'$

**thus**  $\exists Q'. Q \longmapsto (STCal\ Source\ Target)^*\ Q' \wedge (P', Q') \in (indRelRTPO\ TRel)^{-1}$

**proof** (*induct arbitrary: P'*)

**case** (*encR S*)

**assume**  $TargetTerm\ (\llbracket S \rrbracket) \longmapsto (STCal\ Source\ Target)^*\ P'$

**from** *this* **obtain**  $T$  **where**  $A1: T \in T\ P'$  **and**  $A2: \llbracket S \rrbracket \longmapsto Target^*\ T$

**by** (*auto simp add: STCal-steps(2)*)

**from** *os*  $A2$  **obtain**  $S'$  **where**  $A3: S \longmapsto Source^*\ S'$  **and**  $A4: (\llbracket S \rrbracket, T) \in TRel^*$

**by** *blast*

**from**  $A3$  **have**  $SourceTerm\ S \longmapsto (STCal\ Source\ Target)^*\ (SourceTerm\ S')$

**by** (*simp add: STCal-steps(1)*)

**moreover** **have**  $SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$

**proof**  $-$

**from**  $A4$  **have**  $\llbracket S \rrbracket = T \vee (\llbracket S \rrbracket, T) \in TRel^+$



**using** *rtrancl-eq-or-trancl*[of  $\llbracket S \rrbracket T TRel$ ]  
**by** *blast*  
**moreover have**  $A5: SourceTerm S' \lesssim[\cdot]RT < TRel > TargetTerm (\llbracket S \rrbracket)$   
**by** (*simp add: indRelRTPO.encR*)  
**with**  $A1$  **have**  $\llbracket S \rrbracket = T \implies SourceTerm S' \lesssim[\cdot]RT < TRel > P'$   
**by** *simp*  
**moreover have**  $(\llbracket S \rrbracket, T) \in TRel^+ \implies SourceTerm S' \lesssim[\cdot]RT < TRel > P'$   
**proof** –  
**assume**  $(\llbracket S \rrbracket, T) \in TRel^+$   
**hence**  $TargetTerm (\llbracket S \rrbracket) \lesssim[\cdot]RT < TRel > TargetTerm T$   
**by** (*rule transitive-closure-of-TRel-to-indRelRTPO*)  
**with**  $A5$  **have**  $SourceTerm S' \lesssim[\cdot]RT < TRel > TargetTerm T$   
**by** (*rule indRelRTPO.trans*)  
**with**  $A1$  **show**  $SourceTerm S' \lesssim[\cdot]RT < TRel > P'$   
**by** *simp*  
**qed**  
**ultimately show**  $SourceTerm S' \lesssim[\cdot]RT < TRel > P'$   
**by** *blast*  
**qed**  
**hence**  $(P', SourceTerm S') \in (indRelRTPO TRel)^{-1}$   
**by** *simp*  
**ultimately**  
**show**  $\exists Q'. SourceTerm S \mapsto (STCal Source Target)^* Q' \wedge (P', Q') \in (indRelRTPO TRel)^{-1}$   
**by** *blast*  
**next**  
**case** (*source S*)  
**then obtain**  $S'$  **where**  $B1: S' \in S P'$   
**by** (*auto simp add: STCal-steps(1)*)  
**hence**  $(P', P') \in (indRelRTPO TRel)^{-1}$   
**by** (*simp add: indRelRTPO.source*)  
**with** *source*  
**show**  $\exists Q'. SourceTerm S \mapsto (STCal Source Target)^* Q' \wedge (P', Q') \in (indRelRTPO TRel)^{-1}$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $TargetTerm T2 \mapsto (STCal Source Target)^* P'$   
**from this obtain**  $T2'$  **where**  $C1: T2' \in T P'$  **and**  $C2: T2 \mapsto Target^* T2'$   
**by** (*auto simp add: STCal-steps(2)*)  
**assume**  $(T1, T2) \in TRel$   
**hence**  $(T2, T1) \in (TRel^+)^{-1}$   
**by** *simp*  
**with**  $C2$  **sim obtain**  $T1'$  **where**  $C3: T1 \mapsto Target^* T1'$  **and**  $C4: (T2', T1') \in (TRel^+)^{-1}$   
**by** *blast*  
**from**  $C3$  **have**  $TargetTerm T1 \mapsto (STCal Source Target)^* (TargetTerm T1')$   
**by** (*simp add: STCal-steps(2)*)  
**moreover from**  $C4$  **have**  $(T1', T2') \in TRel^+$   
**by** *induct*  
**hence**  $TargetTerm T1' \lesssim[\cdot]RT < TRel > TargetTerm T2'$   
**by** (*rule transitive-closure-of-TRel-to-indRelRTPO*)  
**with**  $C1$  **have**  $(P', TargetTerm T1') \in (indRelRTPO TRel)^{-1}$   
**by** *simp*  
**ultimately**  
**show**  $\exists Q'. TargetTerm T1 \mapsto (STCal Source Target)^* Q' \wedge (P', Q') \in (indRelRTPO TRel)^{-1}$   
**by** *blast*  
**next**  
**case** (*trans P Q R R'*)  
**assume**  $R \mapsto (STCal Source Target)^* R'$   
**and**  $\bigwedge R'. R \mapsto (STCal Source Target)^* R'$   
 $\implies \exists Q'. Q \mapsto (STCal Source Target)^* Q' \wedge (R', Q') \in (indRelRTPO TRel)^{-1}$   
**from this obtain**  $Q'$  **where**  $D1: Q \mapsto (STCal Source Target)^* Q'$   
**and**  $D2: (R', Q') \in (indRelRTPO TRel)^{-1}$

by blast  
 assume  $\bigwedge Q'. Q \mapsto (STCal\ Source\ Target)*\ Q'$   
 $\implies \exists P'. P \mapsto (STCal\ Source\ Target)*\ P' \wedge (Q', P') \in (indRelRTPO\ TRel)^{-1}$   
 with  $D1$  obtain  $P'$  where  $D3: P \mapsto (STCal\ Source\ Target)*\ P'$   
     and  $D4: (Q', P') \in (indRelRTPO\ TRel)^{-1}$   
 by blast  
 from  $D4\ D2$  have  $(R', P') \in (indRelRTPO\ TRel)^{-1}$   
     by (simp add: indRelRTPO.trans[where  $P=P'$  and  $Q=Q'$  and  $R=R'$ ])  
 with  $D3$  show  $\exists P'. P \mapsto (STCal\ Source\ Target)*\ P' \wedge (R', P') \in (indRelRTPO\ TRel)^{-1}$   
 by blast  
 qed  
 qed  
 next  
 have  $\forall S. SourceTerm\ S \lesssim [\cdot]RT<TRel>\ TargetTerm\ ([S])$   
     by (simp add: indRelRTPO.encR)  
 moreover have  $\forall S\ T. SourceTerm\ S \lesssim [\cdot]RT<TRel>\ TargetTerm\ T \longrightarrow ([S], T) \in TRel^*$   
     using indRelRTPO-to-TRel(2)[where  $TRel=TRel$ ] trans-closure-of-TRel-refl-cond  
     by simp  
 moreover  
 assume sim: weak-reduction-simulation  $((indRelRTPO\ TRel)^{-1})\ (STCal\ Source\ Target)$   
 ultimately have operational-sound  $(TRel^*)$   
     using weak-reduction-simulation-impl-OSou[where  $Rel=indRelRTPO\ TRel$  and  $TRel=TRel$ ]  
     by simp  
 moreover from sim have weak-reduction-simulation  $((TRel^+)^{-1})\ Target$   
     using indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev[where  $TRel=TRel$ ]  
     by simp  
 ultimately show operational-sound  $(TRel^*) \wedge$  weak-reduction-simulation  $((TRel^+)^{-1})\ Target$   
     by simp  
 qed  
  
**lemma** (in encoding) OSou-iff-weak-reduction-simulation:  
 fixes  $TRel :: ('procT \times 'procT)\ set$   
 shows (operational-sound  $(TRel^*)$   
      $\wedge$  weak-reduction-simulation  $((TRel^+)^{-1})\ Target$   
      $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel)$   
      $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
      $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
      $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow ([S], T) \in TRel^*)$   
      $\wedge$  weak-reduction-simulation  $(Rel^{-1})\ (STCal\ Source\ Target))$ )  
**proof** (rule iffI, erule conjE)  
 have  $\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in indRelRTPO\ TRel$   
     by (simp add: indRelRTPO.encR)  
 moreover have  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow TargetTerm\ T1 \lesssim [\cdot]RT<TRel>\ TargetTerm\ T2$   
     by (simp add: indRelRTPO.target)  
 moreover have  $\forall T1\ T2. TargetTerm\ T1 \lesssim [\cdot]RT<TRel>\ TargetTerm\ T2 \longrightarrow (T1, T2) \in TRel^+$   
     using indRelRTPO-to-TRel(4)[where  $TRel=TRel$ ]  
     by simp  
 moreover have  $\forall S\ T. SourceTerm\ S \lesssim [\cdot]RT<TRel>\ TargetTerm\ T \longrightarrow ([S], T) \in TRel^*$   
     using indRelRTPO-to-TRel(2)[where  $TRel=TRel$ ] trans-closure-of-TRel-refl-cond  
     by simp  
 moreover assume operational-sound  $(TRel^*)$   
     and weak-reduction-simulation  $((TRel^+)^{-1})\ Target$   
 hence weak-reduction-simulation  $((indRelRTPO\ TRel)^{-1})\ (STCal\ Source\ Target)$   
     using OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation[where  $TRel=TRel$ ]  
     by simp  
 ultimately show  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel)$   
      $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
      $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
      $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow ([S], T) \in TRel^*)$   
      $\wedge$  weak-reduction-simulation  $(Rel^{-1})\ (STCal\ Source\ Target)$   
     by blast

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge weak-reduction-simulation\ (Rel^{-1})\ (STCal\ Source\ Target)$   
**from this obtain Rel where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $A3: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $A4: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $A5: weak-reduction-simulation\ (Rel^{-1})\ (STCal\ Source\ Target)$   
**by blast**  
**from**  $A1\ A4\ A5$  **have** *operational-sound*  $(TRel^*)$   
**using** *weak-reduction-simulation-impl-OSou* [**where**  $Rel=Rel$  **and**  $TRel=TRel$ ]  
**by simp**  
**moreover from**  $A2\ A3\ A5$  **have** *weak-reduction-simulation*  $((TRel^+)^{-1})\ Target$   
**using** *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev* [**where**  $Rel=Rel$  **and**  
 $TRel=TRel$ ]  
**by simp**  
**ultimately show** *operational-sound*  $(TRel^*) \wedge weak-reduction-simulation\ ((TRel^+)^{-1})\ Target$   
**by simp**  
**qed**

An encoding is strongly operational sound modulo a relation  $TRel$  whose inverse is a strong reduction simulation on target terms iff there is a relation, like  $indRelRTPO$ , that relates at least all source terms to their literal translations, includes  $TRel$ , and whose inverse is a strong simulation.

**lemma** (*in encoding*) *strong-reduction-simulation-impl-SOSou*:

**fixes**  $Rel :: ('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc$  *set*  
**and**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $A3: strong-reduction-simulation\ (Rel^{-1})\ (STCal\ Source\ Target)$   
**shows** *strongly-operational-sound*  $(TRel^*)$

**proof** *clarify*

**fix**  $S\ T$   
**from**  $A1$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel^{-1}$   
**by simp**  
**moreover assume**  $\llbracket S \rrbracket \mapsto Target\ T$   
**hence**  $TargetTerm\ (\llbracket S \rrbracket) \mapsto (STCal\ Source\ Target)\ (TargetTerm\ T)$   
**by** (*simp add: STCal-step(2)*)  
**ultimately obtain**  $Q'$  **where**  $A5: SourceTerm\ S \mapsto (STCal\ Source\ Target)\ Q'$   
**and**  $A6: (TargetTerm\ T, Q') \in Rel^{-1}$   
**using**  $A3$   
**by blast**  
**from**  $A5$  **obtain**  $S'$  **where**  $A7: S' \in S\ Q'$  **and**  $A8: S \mapsto Source\ S'$   
**by** (*auto simp add: STCal-step(1)*)  
**from**  $A6$  **have**  $(Q', TargetTerm\ T) \in Rel$   
**by induct**  
**with**  $A2\ A7$  **have**  $(\llbracket S \rrbracket, T) \in TRel^*$   
**by simp**  
**with**  $A8$  **show**  $\exists S'. S \mapsto Source\ S' \wedge (\llbracket S \rrbracket, T) \in TRel^*$   
**by blast**

**qed**

**lemma** (*in encoding*) *SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation*:

**fixes**  $TRel :: ('procT \times 'procT)$  *set*  
**shows** (*strongly-operational-sound*  $(TRel^*)$   
 $\wedge strong-reduction-simulation\ ((TRel^+)^{-1})\ Target$   
 $= strong-reduction-simulation\ ((indRelRTPO\ TRel)^{-1})\ (STCal\ Source\ Target)$ )

**proof** (*rule iffI, erule conjE*)

**assume** *os*: *strongly-operational-sound* ( $TRel^*$ )  
**and** *sim*: *strong-reduction-simulation* ( $(TRel^+)^{-1}$ ) *Target*  
**show** *strong-reduction-simulation* ( $(indRelRTPO\ TRel)^{-1}$ ) (*STCal Source Target*)  
**proof** *clarify*  
**fix**  $P\ Q\ P'$   
**assume**  $Q \lesssim [\cdot]RT < TRel > P$   
**moreover assume**  $P \mapsto (STCal\ Source\ Target)\ P'$   
**ultimately**  
**show**  $\exists Q'. Q \mapsto (STCal\ Source\ Target)\ Q' \wedge (P', Q') \in (indRelRTPO\ TRel)^{-1}$   
**proof** (*induct arbitrary: P'*)  
**case** (*encR S*)  
**assume**  $TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)\ P'$   
**from this obtain**  $T$  **where**  $A1: T \in T\ P'$  **and**  $A2: [S] \mapsto Target\ T$   
**by** (*auto simp add: STCal-step(2)*)  
**from os A2 obtain**  $S'$  **where**  $A3: S \mapsto Source\ S'$  **and**  $A4: ([S'], T) \in TRel^*$   
**by** *blast*  
**from A3 have**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)\ (SourceTerm\ S')$   
**by** (*simp add: STCal-step(1)*)  
**moreover have**  $SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$   
**proof** –  
**from A4 have**  $[S'] = T \vee ([S'], T) \in TRel^+$   
**using** *rtrancl-eq-or-trancl*[of  $[S']\ T\ TRel$ ]  
**by** *blast*  
**moreover have**  $A5: SourceTerm\ S' \lesssim [\cdot]RT < TRel > TargetTerm\ ([S'])$   
**by** (*simp add: indRelRTPO.encR*)  
**with A1 have**  $[S'] = T \implies SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$   
**by** *simp*  
**moreover have**  $([S'], T) \in TRel^+ \implies SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$   
**proof** –  
**assume**  $([S'], T) \in TRel^+$   
**hence**  $TargetTerm\ ([S']) \lesssim [\cdot]RT < TRel > TargetTerm\ T$   
**by** (*rule transitive-closure-of-TRel-to-indRelRTPO*)  
**with A5 have**  $SourceTerm\ S' \lesssim [\cdot]RT < TRel > TargetTerm\ T$   
**by** (*rule indRelRTPO.trans*)  
**with A1 show**  $SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$   
**by** *simp*  
**qed**  
**ultimately show**  $SourceTerm\ S' \lesssim [\cdot]RT < TRel > P'$   
**by** *blast*  
**qed**  
**hence**  $(P', SourceTerm\ S') \in (indRelRTPO\ TRel)^{-1}$   
**by** *simp*  
**ultimately**  
**show**  $\exists Q'. SourceTerm\ S \mapsto (STCal\ Source\ Target)\ Q' \wedge (P', Q') \in (indRelRTPO\ TRel)^{-1}$   
**by** *blast*  
**next**  
**case** (*source S*)  
**then obtain**  $S'$  **where**  $B1: S' \in S\ P'$   
**by** (*auto simp add: STCal-step(1)*)  
**hence**  $(P', P') \in (indRelRTPO\ TRel)^{-1}$   
**by** (*simp add: indRelRTPO.source*)  
**with source**  
**show**  $\exists Q'. SourceTerm\ S \mapsto (STCal\ Source\ Target)\ Q' \wedge (P', Q') \in (indRelRTPO\ TRel)^{-1}$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $TargetTerm\ T2 \mapsto (STCal\ Source\ Target)\ P'$   
**from this obtain**  $T2'$  **where**  $C1: T2' \in T\ P'$  **and**  $C2: T2 \mapsto Target\ T2'$   
**by** (*auto simp add: STCal-step(2)*)  
**assume**  $(T1, T2) \in TRel$   
**hence**  $(T2, T1) \in (TRel^+)^{-1}$

by *simp*  
 with  $C2$  *sim* obtain  $T1'$  where  $C3: T1 \mapsto \text{Target } T1'$  and  $C4: (T2', T1') \in (TRel^+)^{-1}$   
 by *blast*  
 from  $C3$  have  $\text{TargetTerm } T1 \mapsto (\text{STCal Source Target}) (\text{TargetTerm } T1')$   
 by (*simp add: STCal-step(2)*)  
 moreover from  $C4$  have  $(T1', T2') \in TRel^+$   
 by *induct*  
 hence  $\text{TargetTerm } T1' \lesssim [\cdot]RT < TRel > \text{TargetTerm } T2'$   
 by (*rule transitive-closure-of-TRel-to-indRelRTPO*)  
 with  $C1$  have  $(P', \text{TargetTerm } T1') \in (\text{indRelRTPO } TRel)^{-1}$   
 by *simp*  
 ultimately  
 show  $\exists Q'. \text{TargetTerm } T1 \mapsto (\text{STCal Source Target}) Q' \wedge (P', Q') \in (\text{indRelRTPO } TRel)^{-1}$   
 by *blast*  
 next  
 case (*trans P Q R R'*)  
 assume  $R \mapsto (\text{STCal Source Target}) R'$   
 and  $\bigwedge R'. R \mapsto (\text{STCal Source Target}) R'$   
 $\implies \exists Q'. Q \mapsto (\text{STCal Source Target}) Q' \wedge (R', Q') \in (\text{indRelRTPO } TRel)^{-1}$   
 from *this* obtain  $Q'$  where  $D1: Q \mapsto (\text{STCal Source Target}) Q'$   
 and  $D2: (R', Q') \in (\text{indRelRTPO } TRel)^{-1}$   
 by *blast*  
 assume  $\bigwedge Q'. Q \mapsto (\text{STCal Source Target}) Q'$   
 $\implies \exists P'. P \mapsto (\text{STCal Source Target}) P' \wedge (Q', P') \in (\text{indRelRTPO } TRel)^{-1}$   
 with  $D1$  obtain  $P'$  where  $D3: P \mapsto (\text{STCal Source Target}) P'$   
 and  $D4: (Q', P') \in (\text{indRelRTPO } TRel)^{-1}$   
 by *blast*  
 from  $D4$   $D2$  have  $(R', P') \in (\text{indRelRTPO } TRel)^{-1}$   
 by (*simp add: indRelRTPO.trans[where P=P' and Q=Q' and R=R']*)  
 with  $D3$  show  $\exists P'. P \mapsto (\text{STCal Source Target}) P' \wedge (R', P') \in (\text{indRelRTPO } TRel)^{-1}$   
 by *blast*  
 qed  
 qed  
 next  
 have  $\forall S. \text{SourceTerm } S \lesssim [\cdot]RT < TRel > \text{TargetTerm } ([S])$   
 by (*simp add: indRelRTPO.encR*)  
 moreover have  $\forall S T. \text{SourceTerm } S \lesssim [\cdot]RT < TRel > \text{TargetTerm } T \longrightarrow ([S], T) \in TRel^*$   
 using *indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond*  
 by *simp*  
 moreover  
 assume *sim: strong-reduction-simulation*  $((\text{indRelRTPO } TRel)^{-1}) (\text{STCal Source Target})$   
 ultimately have *strongly-operational-sound*  $(TRel^*)$   
 using *strong-reduction-simulation-impl-SOSou[where Rel=indRelRTPO TRel and TRel=TRel]*  
 by *simp*  
 moreover from *sim* have *strong-reduction-simulation*  $((TRel^+)^{-1}) \text{Target}$   
 using *indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev[where TRel=TRel]*  
 by *simp*  
 ultimately  
 show *strongly-operational-sound*  $(TRel^*) \wedge \text{strong-reduction-simulation } ((TRel^+)^{-1}) \text{Target}$   
 by *simp*  
 qed  
 lemma (*in encoding*) *SOSou-iff-strong-reduction-simulation*:  
 fixes  $TRel :: ('procT \times 'procT)$  set  
 shows  $(\text{strongly-operational-sound } (TRel^*) \wedge \text{strong-reduction-simulation } ((TRel^+)^{-1}) \text{Target})$   
 $= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S]))) \in Rel)$   
 $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel^*)$   
 $\wedge \text{strong-reduction-simulation } (Rel^{-1}) (\text{STCal Source Target})$   
 proof (*rule iffI, erule conjE*)

**have**  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } T\text{Rel}$   
**by** (*simp add: indRelRTPO.encR*)  
**moreover have**  $\forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow \text{TargetTerm } T1 \lesssim[\cdot]RT<T\text{Rel}> \text{TargetTerm } T2$   
**by** (*simp add: indRelRTPO.target*)  
**moreover have**  $\forall T1\ T2. \text{TargetTerm } T1 \lesssim[\cdot]RT<T\text{Rel}> \text{TargetTerm } T2 \longrightarrow (T1, T2) \in T\text{Rel}^+$   
**using** *indRelRTPO-to-TRel(4)***[where**  $T\text{Rel}=T\text{Rel}$ **]**  
**by** *simp*  
**moreover have**  $\forall S\ T. \text{SourceTerm } S \lesssim[\cdot]RT<T\text{Rel}> \text{TargetTerm } T \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$   
**using** *indRelRTPO-to-TRel(2)***[where**  $T\text{Rel}=T\text{Rel}$ **]** *trans-closure-of-TRel-refl-cond*  
**by** *simp*  
**moreover assume** *strongly-operational-sound* ( $T\text{Rel}^*$ )  
**and** *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*  
**hence** *strong-reduction-simulation*  $((\text{indRelRTPO } T\text{Rel})^{-1})$  (*STCal Source Target*)  
**using** *SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation***[where**  $T\text{Rel}=T\text{Rel}$ **]**  
**by** *simp*  
**ultimately show**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$   
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$   
 $\wedge$  *strong-reduction-simulation*  $(\text{Rel}^{-1})$  (*STCal Source Target*)  
**by** *blast*  
**next**  
**assume**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$   
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$   
 $\wedge$  *strong-reduction-simulation*  $(\text{Rel}^{-1})$  (*STCal Source Target*)  
**from this obtain** *Rel* **where**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$   
**and**  $A2: \forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$   
**and**  $A3: \forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+$   
**and**  $A4: \forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$   
**and**  $A5: \text{strong-reduction-simulation } (\text{Rel}^{-1})$  (*STCal Source Target*)  
**by** *blast*  
**from**  $A1\ A4\ A5$  **have** *strongly-operational-sound* ( $T\text{Rel}^*$ )  
**using** *strong-reduction-simulation-impl-SOSou***[where**  $\text{Rel}=\text{Rel}$  **and**  $T\text{Rel}=T\text{Rel}$ **]**  
**by** *simp*  
**moreover from**  $A2\ A3\ A5$  **have** *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*  
**using** *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev***[where**  $\text{Rel}=\text{Rel}$  **and**  
 $T\text{Rel}=T\text{Rel}$ **]**  
**by** *simp*  
**ultimately**  
**show** *strongly-operational-sound* ( $T\text{Rel}^*$ )  $\wedge$  *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*  
**by** *simp*  
**qed**

**lemma** (*in encoding*) *SOSou-modulo-TRel-iff-strong-reduction-simulation*:  
**shows**  $(\exists T\text{Rel}. \text{strongly-operational-sound } (T\text{Rel}^*)$   
 $\wedge$  *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*)  
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}^=)$   
 $\wedge$  *strong-reduction-simulation*  $(\text{Rel}^{-1})$  (*STCal Source Target*))

**proof** (*rule iffI*)  
**assume**  $\exists T\text{Rel}. \text{strongly-operational-sound } (T\text{Rel}^*)$   
 $\wedge$  *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*  
**from this obtain**  $T\text{Rel}$  **where** *strongly-operational-sound* ( $T\text{Rel}^*$ )  
**and** *strong-reduction-simulation*  $((T\text{Rel}^+)^{-1})$  *Target*  
**by** *blast*  
**hence** *strong-reduction-simulation*  $((\text{indRelRTPO } T\text{Rel})^{-1})$  (*STCal Source Target*)  
**using** *SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation***[where**  $T\text{Rel}=T\text{Rel}$ **]**  
**by** *simp*  
**moreover have**  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } T\text{Rel}$

by (*simp add: indRelRTPO.encR*)  
**moreover have**  $\forall S T. (SourceTerm S, TargetTerm T) \in indRelRTPO TRel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in (indRelRTPO TRel)^=$   
 using *indRelRTPO-relates-source-target*[**where**  $TRel = TRel$ ]  
 by *simp*  
**ultimately show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=)$   
 $\wedge strong\text{-reduction-simulation } (Rel^{-1}) (STCal Source Target)$   
 by *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=)$   
 $\wedge strong\text{-reduction-simulation } (Rel^{-1}) (STCal Source Target)$   
**from this obtain** *Rel* **where**  $A1: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$   
**and**  $A2: (\forall S T. (SourceTerm S, TargetTerm T) \in Rel$   
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=)$   
**and**  $A3: strong\text{-reduction-simulation } (Rel^{-1}) (STCal Source Target)$   
 by *blast*  
**from**  $A2$  **obtain** *TRel* **where**  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$   
**and**  $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
 using *target-relation-from-source-target-relation*[**where**  $Rel = Rel$ ]  
 by *blast*  
**with**  $A1 A3$   
**have** *strongly-operational-sound* ( $TRel^*$ )  $\wedge strong\text{-reduction-simulation } ((TRel^+)^{-1}) Target$   
 using *SOSou-iff-strong-reduction-simulation*[**where**  $TRel = TRel$ ]  
 by *blast*  
**thus**  $\exists TRel. strongly\text{-operational-sound } (TRel^*) \wedge strong\text{-reduction-simulation } ((TRel^+)^{-1}) Target$   
 by *blast*  
**qed**

## 8.5 Weak Operational Correspondence vs Correspondence Similarity

If there exists a relation that relates at least all source terms and their literal translations, includes  $TRel$ , and is a correspondence simulation then the encoding is weakly operational corresponding w.r.t.  $TRel$ .

**lemma** (*in encoding*) *weak-reduction-correspondence-simulation-impl-WOC*:

**fixes**  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc$  *set*  
**and**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes** *enc*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$   
**and** *tRel*:  $(\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
**and** *cs*: *weak-reduction-correspondence-simulation*  $Rel (STCal Source Target)$   
**shows** *weakly-operational-corresponding* ( $TRel^*$ )  
**proof**  
**from** *enc tRel cs* **show** *operational-complete* ( $TRel^*$ )  
 using *weak-reduction-simulation-impl-OCom*[**where**  $TRel = TRel$ ]  
 by *simp*  
**next**  
**show** *weakly-operational-sound* ( $TRel^*$ )  
**proof** *clarify*  
**fix**  $S T$   
**from** *enc* **have**  $(SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$   
 by *simp*  
**moreover assume**  $\llbracket S \rrbracket \longmapsto Target^* T$   
**hence**  $TargetTerm (\llbracket S \rrbracket) \longmapsto (STCal Source Target)^* (TargetTerm T)$   
 by (*simp add: STCal-steps(2)*)  
**ultimately obtain**  $P' Q'$  **where**  $A1: SourceTerm S \longmapsto (STCal Source Target)^* P'$   
**and**  $A2: TargetTerm T \longmapsto (STCal Source Target)^* Q'$  **and**  $A3: (P', Q') \in Rel$

```

    using cs
  by blast
  from A1 obtain S' where A4: S' ∈ S P' and A5: S ⟶ Source* S'
  by (auto simp add: STCal-steps(1))
  from A2 obtain T' where A6: T' ∈ T Q' and A7: T ⟶ Target* T'
  by (auto simp: STCal-steps(2))
  from tRel A3 A4 A6 have (⟦S⟧, T') ∈ TRel*
  by simp
  with A5 A7 show ∃ S' T'. S ⟶ Source* S' ∧ T ⟶ Target* T' ∧ (⟦S⟧, T') ∈ TRel*
  by blast
qed
qed

```

An encoding is weakly operational corresponding w.r.t. a correspondence simulation on target terms  $TRel$  iff there exists a relation, like  $indRelRTPO$ , that relates at least all source terms and their literal translations, includes  $TRel$ , and is a correspondence simulation.

**lemma** (in *encoding*) *WOC-iff-indRelRTPO-is-reduction-correspondence-simulation*:

**fixes**  $TRel :: ('procT \times 'procT) \text{ set}$

**shows** (*weakly-operational-corresponding* ( $TRel^*$ ))

$\wedge$  *weak-reduction-correspondence-simulation* ( $TRel^+$ ) *Target*)

$=$  *weak-reduction-correspondence-simulation* ( $indRelRTPO \ TRel$ ) ( $STCal \ Source \ Target$ )

**proof** (*rule iffI, erule conjE*)

**assume** *woc: weakly-operational-corresponding* ( $TRel^*$ )

**and** *csi: weak-reduction-correspondence-simulation* ( $TRel^+$ ) *Target*

**show** *weak-reduction-correspondence-simulation* ( $indRelRTPO \ TRel$ ) ( $STCal \ Source \ Target$ )

**proof**

**from** *woc csi* **show** *sim: weak-reduction-simulation* ( $indRelRTPO \ TRel$ ) ( $STCal \ Source \ Target$ )

**using** *OCom-iff-indRelRTPO-is-weak-reduction-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**show**  $\forall P \ Q \ Q'. P \lesssim [\cdot]RT < TRel > Q \wedge Q \mapsto (STCal \ Source \ Target)^* Q'$

$\longrightarrow (\exists P'' \ Q''. P \mapsto (STCal \ Source \ Target)^* P'' \wedge Q' \mapsto (STCal \ Source \ Target)^* Q''$

$\wedge P'' \lesssim [\cdot]RT < TRel > Q'')$

**proof** *clarify*

**fix**  $P \ Q \ Q'$

**assume**  $P \lesssim [\cdot]RT < TRel > Q$  **and**  $Q \mapsto (STCal \ Source \ Target)^* Q'$

**thus**  $\exists P'' \ Q''. P \mapsto (STCal \ Source \ Target)^* P'' \wedge Q' \mapsto (STCal \ Source \ Target)^* Q''$

$\wedge P'' \lesssim [\cdot]RT < TRel > Q''$

**proof** (*induct arbitrary: Q'*)

**case** (*encR S*)

**assume**  $TargetTerm \ (\llbracket S \rrbracket) \mapsto (STCal \ Source \ Target)^* Q'$

**from** *this* **obtain**  $T$  **where**  $A1: T \in T \ Q'$  **and**  $A2: \llbracket S \rrbracket \mapsto Target^* T$

**by** (*auto simp add: STCal-steps(2)*)

**from**  $A2$  *woc* **obtain**  $S' \ T'$  **where**  $A3: S \mapsto Source^* S'$  **and**  $A4: T \mapsto Target^* T'$

**and**  $A5: (\llbracket S \rrbracket, T') \in TRel^*$

**by** *blast*

**from**  $A3$  **have**  $SourceTerm \ S \mapsto (STCal \ Source \ Target)^* (SourceTerm \ S')$

**by** (*simp add: STCal-steps(1)*)

**moreover** **from**  $A4$  **have**  $TargetTerm \ T \mapsto (STCal \ Source \ Target)^* (TargetTerm \ T')$

**by** (*simp add: STCal-steps(2)*)

**moreover** **have**  $SourceTerm \ S' \lesssim [\cdot]RT < TRel > TargetTerm \ T'$

**proof** –

**have**  $A6: SourceTerm \ S' \lesssim [\cdot]RT < TRel > TargetTerm \ (\llbracket S \rrbracket)$

**by** (*rule indRelRTPO.encR*)

**from**  $A5$  **have**  $\llbracket S \rrbracket = T' \vee (\llbracket S \rrbracket, T') \in TRel^+$

**using** *rtrancl-eq-or-trancl*[*of*  $\llbracket S \rrbracket \ T' \ TRel$ ]

**by** *blast*

**moreover** **from**  $A6$  **have**  $\llbracket S \rrbracket = T' \implies SourceTerm \ S' \lesssim [\cdot]RT < TRel > TargetTerm \ T'$

**by** *simp*

**moreover** **have**  $(\llbracket S \rrbracket, T') \in TRel^+ \implies SourceTerm \ S' \lesssim [\cdot]RT < TRel > TargetTerm \ T'$

**proof** –



**assume** ( $\llbracket S \rrbracket, T' \in TRel^+$   
**hence**  $TargetTerm \llbracket S \rrbracket \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T'$   
**by** (*simp add: transitive-closure-of-TRel-to-indRelRTPO*[**where**  $TRel = TRel$ ])  
**with**  $A6$  **show**  $SourceTerm S' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T'$   
**by** (*rule indRelRTPO.trans*)  
**qed**  
**ultimately show**  $SourceTerm S' \lesssim \llbracket \cdot \rrbracket RT < TRel > TargetTerm T'$   
**by** *blast*  
**qed**  
**ultimately show**  $\exists P'' Q''. SourceTerm S \mapsto (STCal Source Target)^* P''$   
 $\wedge Q' \mapsto (STCal Source Target)^* Q'' \wedge P'' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q''$   
**using**  $A1$   
**by** *blast*  
**next**  
**case** (*source S*)  
**assume**  $B1: SourceTerm S \mapsto (STCal Source Target)^* Q'$   
**moreover have**  $Q' \mapsto (STCal Source Target)^* Q'$   
**by** (*rule steps-refl*)  
**moreover from**  $B1$  **obtain**  $S'$  **where**  $S' \in S Q'$   
**by** (*auto simp add: STCal-steps(1)*)  
**hence**  $Q' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q'$   
**by** (*simp add: indRelRTPO.source*)  
**ultimately show**  $\exists P'' Q''. SourceTerm S \mapsto (STCal Source Target)^* P''$   
 $\wedge Q' \mapsto (STCal Source Target)^* Q'' \wedge P'' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q''$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $TargetTerm T2 \mapsto (STCal Source Target)^* Q'$   
**from this obtain**  $T2'$  **where**  $C1: T2' \in T Q'$  **and**  $C2: T2 \mapsto Target^* T2'$   
**by** (*auto simp add: STCal-steps(2)*)  
**assume**  $(T1, T2) \in TRel$   
**hence**  $(T1, T2) \in TRel^+$   
**by** *simp*  
**with**  $C2$  **csi obtain**  $T1' T2''$  **where**  $C3: T1 \mapsto Target^* T1'$  **and**  $C4: T2' \mapsto Target^* T2''$   
**and**  $C5: (T1', T2'') \in TRel^+$   
**by** *blast*  
**from**  $C3$  **have**  $TargetTerm T1 \mapsto (STCal Source Target)^* (TargetTerm T1')$   
**by** (*simp add: STCal-steps(2)*)  
**moreover from**  $C1 C4$  **have**  $Q' \mapsto (STCal Source Target)^* (TargetTerm T2'')$   
**by** (*simp add: STCal-steps(2)*)  
**moreover from**  $C5$  **have**  $TargetTerm T1' \lesssim \llbracket \cdot \rrbracket RT < TRel > (TargetTerm T2'')$   
**by** (*simp add: transitive-closure-of-TRel-to-indRelRTPO*)  
**ultimately show**  $\exists P'' Q''. TargetTerm T1 \mapsto (STCal Source Target)^* P''$   
 $\wedge Q' \mapsto (STCal Source Target)^* Q'' \wedge P'' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q''$   
**by** *blast*  
**next**  
**case** (*trans P Q R R'*)  
**assume**  $R \mapsto (STCal Source Target)^* R'$   
**and**  $\bigwedge R'. R \mapsto (STCal Source Target)^* R' \implies \exists Q'' R''. Q \mapsto (STCal Source Target)^* Q''$   
 $\wedge R' \mapsto (STCal Source Target)^* R'' \wedge Q'' \lesssim \llbracket \cdot \rrbracket RT < TRel > R''$   
**and**  $\bigwedge Q'. Q \mapsto (STCal Source Target)^* Q' \implies \exists P'' Q''. P \mapsto (STCal Source Target)^* P''$   
 $\wedge Q' \mapsto (STCal Source Target)^* Q'' \wedge P'' \lesssim \llbracket \cdot \rrbracket RT < TRel > Q''$   
**moreover have** *trans* (*indRelRTPO TRel*)  
**using** *indRelRTPO.trans*  
**unfolding** *trans-def*  
**by** *blast*  
**ultimately show** *?case*  
**using** *sim reduction-correspondence-simulation-condition-trans*[**where**  $P=P$  **and**  
 $Rel=indRelRTPO TRel$  **and**  $Cal=STCal Source Target$  **and**  $Q=Q$  **and**  $R=R$ ]  
**by** *blast*  
**qed**

**qed**  
**qed**  
**next**  
**assume** *csi*: weak-reduction-correspondence-simulation (*indRelRTPO* *TRel*) (*STCal* *Source* *Target*)  
**show** weakly-operational-corresponding (*TRel*<sup>\*</sup>)  
 $\wedge$  weak-reduction-correspondence-simulation (*TRel*<sup>+</sup>) *Target*  
**proof**  
**have**  $\forall S$ . *SourceTerm* *S*  $\lesssim[\cdot]RT<TRel>$  *TargetTerm* ( $\llbracket S \rrbracket$ )  
**by** (*simp* *add*: *indRelRTPO.encR*)  
**moreover have**  $\forall S T$ . *SourceTerm* *S*  $\lesssim[\cdot]RT<TRel>$  *TargetTerm* *T*  $\longrightarrow$  ( $\llbracket S \rrbracket$ , *T*)  $\in$  *TRel*<sup>\*</sup>  
**using** *indRelRTPO-to-TRel(2)*[**where** *TRel=TRel*] *trans-closure-of-TRel-refl-cond*  
**by** *simp*  
**ultimately show** weakly-operational-corresponding (*TRel*<sup>\*</sup>)  
**using** *weak-reduction-correspondence-simulation-impl-WOC*[**where** *Rel=indRelRTPO TRel* **and** *TRel=TRel*] *csi*  
**by** *simp*  
**next**  
**from** *csi* **show** weak-reduction-correspondence-simulation (*TRel*<sup>+</sup>) *Target*  
**using** *indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation*[**where** *TRel=TRel*]  
**by** *simp*  
**qed**  
**qed**

**lemma** (**in** *encoding*) *WOC-iff-reduction-correspondence-simulation*:

**fixes** *TRel* :: ('*procT*  $\times$  '*procT*) *set*

**shows** (weakly-operational-corresponding (*TRel*<sup>\*</sup>)

$\wedge$  weak-reduction-correspondence-simulation (*TRel*<sup>+</sup>) *Target*)  
 $=$  ( $\exists$  *Rel*. ( $\forall S$ . (*SourceTerm* *S*, *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *Rel*)  
 $\wedge$  ( $\forall T1 T2$ . (*T1*, *T2*)  $\in$  *TRel*  $\longrightarrow$  (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*)  
 $\wedge$  ( $\forall T1 T2$ . (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*  $\longrightarrow$  (*T1*, *T2*)  $\in$  *TRel*<sup>+</sup>)  
 $\wedge$  ( $\forall S T$ . (*SourceTerm* *S*, *TargetTerm* *T*)  $\in$  *Rel*  $\longrightarrow$  ( $\llbracket S \rrbracket$ , *T*)  $\in$  *TRel*<sup>\*</sup>)  
 $\wedge$  weak-reduction-correspondence-simulation *Rel* (*STCal* *Source* *Target*))

**proof** (*rule iffI*, *erule conjE*)

**have**  $\forall S$ . (*SourceTerm* *S*, *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *indRelRTPO* *TRel*

**by** (*simp* *add*: *indRelRTPO.encR*)

**moreover have**  $\forall T1 T2$ . (*T1*, *T2*)  $\in$  *TRel*  $\longrightarrow$  *TargetTerm* *T1*  $\lesssim[\cdot]RT<TRel>$  *TargetTerm* *T2*

**by** (*simp* *add*: *indRelRTPO.target*)

**moreover have**  $\forall T1 T2$ . *TargetTerm* *T1*  $\lesssim[\cdot]RT<TRel>$  *TargetTerm* *T2*  $\longrightarrow$  (*T1*, *T2*)  $\in$  *TRel*<sup>+</sup>

**using** *indRelRTPO-to-TRel(4)*[**where** *TRel=TRel*]

**by** *simp*

**moreover have**  $\forall S T$ . *SourceTerm* *S*  $\lesssim[\cdot]RT<TRel>$  *TargetTerm* *T*  $\longrightarrow$  ( $\llbracket S \rrbracket$ , *T*)  $\in$  *TRel*<sup>\*</sup>

**using** *indRelRTPO-to-TRel(2)*[**where** *TRel=TRel*] *trans-closure-of-TRel-refl-cond*

**by** *simp*

**moreover assume** weakly-operational-corresponding (*TRel*<sup>\*</sup>)

**and** weak-reduction-correspondence-simulation (*TRel*<sup>+</sup>) *Target*

**hence** weak-reduction-correspondence-simulation (*indRelRTPO* *TRel*) (*STCal* *Source* *Target*)

**using** *WOC-iff-indRelRTPO-is-reduction-correspondence-simulation*[**where** *TRel=TRel*]

**by** *simp*

**ultimately show**  $\exists$  *Rel*. ( $\forall S$ . (*SourceTerm* *S*, *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *Rel*)

$\wedge$  ( $\forall T1 T2$ . (*T1*, *T2*)  $\in$  *TRel*  $\longrightarrow$  (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*)

$\wedge$  ( $\forall T1 T2$ . (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*  $\longrightarrow$  (*T1*, *T2*)  $\in$  *TRel*<sup>+</sup>)

$\wedge$  ( $\forall S T$ . (*SourceTerm* *S*, *TargetTerm* *T*)  $\in$  *Rel*  $\longrightarrow$  ( $\llbracket S \rrbracket$ , *T*)  $\in$  *TRel*<sup>\*</sup>)

$\wedge$  weak-reduction-correspondence-simulation *Rel* (*STCal* *Source* *Target*)

**by** *blast*

**next**

**assume**  $\exists$  *Rel*. ( $\forall S$ . (*SourceTerm* *S*, *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *Rel*)

$\wedge$  ( $\forall T1 T2$ . (*T1*, *T2*)  $\in$  *TRel*  $\longrightarrow$  (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*)

$\wedge$  ( $\forall T1 T2$ . (*TargetTerm* *T1*, *TargetTerm* *T2*)  $\in$  *Rel*  $\longrightarrow$  (*T1*, *T2*)  $\in$  *TRel*<sup>+</sup>)

$\wedge$  ( $\forall S T$ . (*SourceTerm* *S*, *TargetTerm* *T*)  $\in$  *Rel*  $\longrightarrow$  ( $\llbracket S \rrbracket$ , *T*)  $\in$  *TRel*<sup>\*</sup>)

$\wedge$  weak-reduction-correspondence-simulation *Rel* (*STCal* *Source* *Target*)

**from** *this* **obtain** *Rel* **where** *A1*:  $\forall S$ . (*SourceTerm* *S*, *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$  *Rel*

**and**  $A2: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $A3: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $A4: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $A5: \text{weak-reduction-correspondence-simulation } Rel\ (STCal\ Source\ Target)$   
**by** *blast*  
**from**  $A1\ A4\ A5$  **have** *weakly-operational-corresponding*  $(TRel^*)$   
**using** *weak-reduction-correspondence-simulation-impl-WOC*[**where**  $Rel=Rel$  **and**  $TRel=TRel$ ]  
**by** *simp*  
**moreover from**  $A2\ A3\ A5$  **have** *weak-reduction-correspondence-simulation*  $(TRel^+)$  *Target*  
**using** *rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation*  
**by** *simp*  
**ultimately show** *weakly-operational-corresponding*  $(TRel^*)$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $(TRel^+)$  *Target*  
**by** *simp*  
**qed**

**lemma** *rel-includes-TRel-modulo-preorder*:

**fixes**  $Rel :: ('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc$  *set*  
**and**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes** *transT*: *trans*  $TRel$   
**shows**  $(\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $= (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$

**proof** (*rule iffI*, *erule conjE*)

**assume**  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**with** *transT* **show**  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**using** *trancl-id*[*of*  $TRel$ ]  
**by** *blast*

**next**

**assume**  $A: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**hence**  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**by** *simp*  
**moreover from** *transT*  $A$   
**have**  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**using** *trancl-id*[*of*  $TRel$ ]  
**by** *blast*  
**ultimately show**  $(\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
**by** *simp*

**qed**

**lemma** (*in encoding*) *WOC-wrt-preorder-iff-reduction-correspondence-simulation*:

**fixes**  $TRel :: ('procT \times 'procT)$  *set*  
**shows** (*weakly-operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $TRel\ Target$ )  
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *preorder*  $Rel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $Rel\ (STCal\ Source\ Target))$

**proof** (*rule iffI*, *erule conjE*, *erule conjE*, *erule conjE*)

**assume**  $A1: \text{operational-complete } TRel$  **and**  $A2: \text{weakly-operational-sound } TRel$   
**and**  $A3: \text{preorder } TRel$  **and**  $A4: \text{weak-reduction-correspondence-simulation } TRel\ Target$   
**from**  $A3$  **have**  $A5: TRel^+ = TRel$   
**using** *trancl-id*[*of*  $TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *blast*  
**with**  $A3$  **have**  $TRel^* = TRel$   
**using** *trancl-id*[*of*  $TRel$ ] *reflcl-trancl*[*of*  $TRel$ ]  
**unfolding** *preorder-on-def* *refl-on-def*

```

  by auto
with A1 A2 have weakly-operational-corresponding (TRel*)
  by simp
moreover from A4 A5 have weak-reduction-correspondence-simulation (TRel+) Target
  by simp
ultimately
have weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)
  using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where TRel=TRel]
  by blast
moreover have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO TRel}$ 
  by (simp add: indRelRTPO.encR)
moreover
have TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelRTPO TRel}\}$ 
proof auto
  fix TP TQ
  assume (TP, TQ)  $\in$  TRel
  thus TargetTerm TP  $\lesssim$   $\llbracket \cdot \rrbracket$  RT < TRel > TargetTerm TQ
    by (rule indRelRTPO.target)
next
  fix TP TQ
  assume TargetTerm TP  $\lesssim$   $\llbracket \cdot \rrbracket$  RT < TRel > TargetTerm TQ
  with A3 show (TP, TQ)  $\in$  TRel
    using indRelRTPO-to-TRel(4)[where TRel=TRel] trancl-id[of TRel]
    unfolding preorder-on-def
    by blast
qed
moreover from A3
have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO TRel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^+$ 
  using indRelRTPO-to-TRel(2)[where TRel=TRel] reflcl-trancl[of TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  unfolding preorder-on-def refl-on-def
  by blast
with A3 have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO TRel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}$ 
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
moreover from A3 have refl (indRelRTPO TRel)
  using indRelRTPO-refl[of TRel]
  unfolding preorder-on-def
  by simp
moreover have trans (indRelRTPO TRel)
  using indRelRTPO.trans
  unfolding trans-def
  by blast
ultimately show  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$ 
   $\wedge$  TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$ 
   $\wedge$   $(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$ 
   $\wedge$  preorder Rel
   $\wedge$  weak-reduction-correspondence-simulation Rel (STCal Source Target)
  unfolding preorder-on-def
  by blast
next
assume  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$ 
   $\wedge$  TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$ 
   $\wedge$   $(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$ 
   $\wedge$  preorder Rel
   $\wedge$  weak-reduction-correspondence-simulation Rel (STCal Source Target)
from this obtain Rel where B1:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$ 
and B2: TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$ 
and B3:  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}$  and B4: preorder Rel
and B5: weak-reduction-correspondence-simulation Rel (STCal Source Target)

```

```

  by blast
from B2 B4 have B6: refl TRel
  unfolding preorder-on-def refl-on-def
  by blast
from B2 B4 have B7: trans TRel
  unfolding trans-def preorder-on-def
  by blast
hence B8: TRel+ = TRel
  using trancl-id[of TRel]
  by simp
with B6 have TRel* = TRel
  using reflcl-trancl[of TRel]
  unfolding refl-on-def
  by blast
with B1 B3 B5 have weakly-operational-corresponding TRel
  using weak-reduction-correspondence-simulation-impl-WOC[where Rel=Rel and TRel=TRel]
  by simp
moreover from B6 B7 have preorder TRel
  unfolding preorder-on-def
  by blast
moreover from B2 B5 B7 B8 have weak-reduction-correspondence-simulation TRel Target
  using rel-includes-TRel-modulo-preorder[where Rel=Rel and TRel=TRel]
  rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where
    Rel=Rel and TRel=TRel]
  by fast
ultimately show weakly-operational-corresponding TRel  $\wedge$  preorder TRel
   $\wedge$  weak-reduction-correspondence-simulation TRel Target
  by blast
qed

```

## 8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity

An encoding is operational corresponding w.r.t a weak bisimulation on target terms TRel iff there exists a relation, like indRelRTPO, that relates at least all source terms and their literal translations, includes TRel, and is a weak bisimulation. Thus this variant of operational correspondence ensures that source terms and their translations are weak bisimilar.

**lemma** (in encoding) OC-iff-indRelRTPO-is-weak-reduction-bisimulation:

**fixes** TRel :: ('procT  $\times$  'procT) set

**shows** (operational-corresponding (TRel\*))

$\wedge$  weak-reduction-bisimulation (TRel<sup>+</sup>) Target

= weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

**proof** (rule iffI, erule conjE)

**assume** occur: operational-corresponding (TRel\*)

**and** bisim: weak-reduction-bisimulation (TRel<sup>+</sup>) Target

**hence** weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)

**using** OCom-iff-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]

**by** simp

**moreover from** bisim **have** weak-reduction-simulation ((TRel<sup>+</sup>)<sup>-1</sup>) Target

**using** weak-reduction-bisimulations-impl-inverse-is-simulation[where Rel=TRel<sup>+</sup>]

**by** simp

**with** occur **have** weak-reduction-simulation ((indRelRTPO TRel)<sup>-1</sup>) (STCal Source Target)

**using** OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]

**by** simp

**ultimately show** weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

**using** weak-reduction-simulations-impl-bisimulation[where Rel=indRelRTPO TRel]

**by** simp

**next**

**assume** bisim: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

**hence** operational-complete (TRel\*)  $\wedge$  weak-reduction-simulation (TRel<sup>+</sup>) Target

**using** OCom-iff-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]

by *simp*  
 moreover from *bisim*  
 have *weak-reduction-simulation*  $((\text{indRelRTPO } T\text{Rel})^{-1})$   $(\text{STCal Source Target})$   
   using *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where**  $\text{Rel}=\text{indRelRTPO } T\text{Rel}$ ]  
   by *simp*  
 hence *operational-sound*  $(T\text{Rel}^*) \wedge \text{weak-reduction-simulation } ((T\text{Rel}^+)^{-1})$  *Target*  
   using *OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation*[**where**  $T\text{Rel}=T\text{Rel}$ ]  
   by *simp*  
 ultimately show *operational-corresponding*  $(T\text{Rel}^*) \wedge \text{weak-reduction-bisimulation } (T\text{Rel}^+)$  *Target*  
   using *weak-reduction-simulations-impl-bisimulation*[**where**  $\text{Rel}=T\text{Rel}^+$ ]  
   by *simp*  
 qed

**lemma** (in *encoding*) *OC-iff-weak-reduction-bisimulation*:

fixes  $T\text{Rel} :: ('procT \times 'procT)$  set

shows  $(\text{operational-corresponding } (T\text{Rel}^*) \wedge \text{weak-reduction-bisimulation } (T\text{Rel}^+))$  *Target*

$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}))$   
 $\wedge (\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$   
 $\wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal Source Target})$

**proof** (rule *iffI*, erule *conjE*)

have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } T\text{Rel}$

  by (*simp add: indRelRTPO.encR*)

moreover have  $\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow \text{TargetTerm } T1 \lesssim[\cdot]_{RT<T\text{Rel}>} \text{TargetTerm } T2$

  by (*simp add: indRelRTPO.target*)

moreover have  $\forall T1 T2. \text{TargetTerm } T1 \lesssim[\cdot]_{RT<T\text{Rel}>} \text{TargetTerm } T2 \longrightarrow (T1, T2) \in T\text{Rel}^+$

  using *indRelRTPO-to-TRel(4)*[**where**  $T\text{Rel}=T\text{Rel}$ ]

  by *simp*

moreover have  $\forall S T. \text{SourceTerm } S \lesssim[\cdot]_{RT<T\text{Rel}>} \text{TargetTerm } T \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$

  using *indRelRTPO-to-TRel(2)*[**where**  $T\text{Rel}=T\text{Rel}$ ] *trans-closure-of-TRel-refl-cond*

  by *simp*

moreover assume *operational-corresponding*  $(T\text{Rel}^*)$

  and *weak-reduction-bisimulation*  $(T\text{Rel}^+)$  *Target*

hence *weak-reduction-bisimulation*  $(\text{indRelRTPO } T\text{Rel})$   $(\text{STCal Source Target})$

  using *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where**  $T\text{Rel}=T\text{Rel}$ ]

  by *simp*

ultimately show  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge (\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$   
 $\wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal Source Target})$

  by *blast*

**next**

assume  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge (\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$

$\wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal Source Target})$

from *this* obtain *Rel* **where**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

  and  $A2: \forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$

  and  $A3: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+$

  and  $A4: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$

  and  $A5: \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal Source Target})$

  by *blast*

hence *operational-complete*  $(T\text{Rel}^*)$

$\wedge \text{weak-reduction-simulation } (T\text{Rel}^+)$  *Target*

  using *OCom-iff-weak-reduction-simulation*[**where**  $T\text{Rel}=T\text{Rel}$ ]

  by *blast*

moreover from  $A5$  have *weak-reduction-simulation*  $(\text{Rel}^{-1})$   $(\text{STCal Source Target})$

  using *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where**  $\text{Rel}=\text{Rel}$ ]

by *simp*  
 with  $A1\ A2\ A3\ A4$  **have** *operational-sound* ( $TRel^*$ )  
      $\wedge$  *weak-reduction-simulation*  $((TRel^+)^{-1})\ Target$   
     **using** *OSou-iff-weak-reduction-simulation*[**where**  $TRel = TRel$ ]  
     by *blast*  
 ultimately **show** *operational-corresponding* ( $TRel^*$ )  
      $\wedge$  *weak-reduction-bisimulation* ( $TRel^+$ )  $Target$   
     **using** *weak-reduction-simulations-impl-bisimulation*[**where**  $Rel = TRel^+$ ]  
     by *simp*  
 qed

**lemma** (in *encoding*) *OC-wrt-preorder-iff-weak-reduction-bisimulation*:

**fixes**  $TRel :: ('procT \times 'procT)\ set$

**shows** (*operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$

$\wedge$  *weak-reduction-bisimulation*  $TRel\ Target$ )

$= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$

$\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

$\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$

$\wedge$  *preorder*  $Rel$

$\wedge$  *weak-reduction-bisimulation*  $Rel\ (STCal\ Source\ Target)$ )

**proof** (*rule iffI*, *erule conjE*, *erule conjE*, *erule conjE*)

**assume**  $A1$ : *operational-complete*  $TRel$  **and**  $A2$ : *operational-sound*  $TRel$

**and**  $A3$ : *preorder*  $TRel$  **and**  $A4$ : *weak-reduction-bisimulation*  $TRel\ Target$

**from**  $A3$  **have**  $A5$ :  $TRel^+ = TRel$

**using** *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def*

by *blast*

**with**  $A3$  **have**  $TRel^* = TRel$

**using** *reflcl-trancl*[of  $TRel$ ]

**unfolding** *preorder-on-def refl-on-def*

by *blast*

**with**  $A1\ A2$  **have** *operational-corresponding* ( $TRel^*$ )

by *simp*

**moreover from**  $A4\ A5$  **have** *weak-reduction-bisimulation* ( $TRel^+$ )  $Target$

by *simp*

**ultimately**

**have** *weak-reduction-bisimulation* (*indRelRTPO*  $TRel$ ) ( $STCal\ Source\ Target$ )

**using** *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]

by *blast*

**moreover have**  $\forall S. SourceTerm\ S \lesssim [\cdot]RT < TRel > TargetTerm\ (\llbracket S \rrbracket)$

by (*simp add: indRelRTPO.encR*)

**moreover**

**have**  $TRel = \{(T1, T2). TargetTerm\ T1 \lesssim [\cdot]RT < TRel > TargetTerm\ T2\}$

**proof** *auto*

**fix**  $TP\ TQ$

**assume**  $(TP, TQ) \in TRel$

**thus**  $TargetTerm\ TP \lesssim [\cdot]RT < TRel > TargetTerm\ TQ$

by (*rule indRelRTPO.target*)

**next**

**fix**  $TP\ TQ$

**assume**  $TargetTerm\ TP \lesssim [\cdot]RT < TRel > TargetTerm\ TQ$

**with**  $A3$  **show**  $(TP, TQ) \in TRel$

**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ] *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def*

by *blast*

qed

**moreover from**  $A3$

**have**  $\forall S\ T. SourceTerm\ S \lesssim [\cdot]RT < TRel > TargetTerm\ T \longrightarrow (\llbracket S \rrbracket, T) \in TRel^+$

**using** *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ] *reflcl-trancl*[of  $TRel$ ]

*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]

**unfolding** *preorder-on-def refl-on-def*

by *auto*  
 with  $A3$  have  $\forall S T. \text{SourceTerm } S \lesssim_{[\cdot]RT} \langle TRel \rangle \text{TargetTerm } T \longrightarrow ([S], T) \in TRel$   
   using *trancl-id*[of  $TRel$ ]  
   unfolding *preorder-on-def*  
 by *blast*  
 moreover from  $A3$  have *refl* (*indRelRTPO*  $TRel$ )  
   unfolding *preorder-on-def*  
   by (*simp add: indRelRTPO-refl*)  
 moreover have *trans* (*indRelRTPO*  $TRel$ )  
   using *indRelRTPO.trans*  
   unfolding *trans-def*  
 by *blast*  
 ultimately show  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel)$   
    $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$   
    $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel)$   
    $\wedge \text{preorder } Rel$   
    $\wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)}$   
   unfolding *preorder-on-def*  
 by *blast*  
 next  
 assume  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel)$   
    $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$   
    $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel)$   
    $\wedge \text{preorder } Rel$   
    $\wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)}$   
 from *this* obtain *Rel* where  $B1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel$   
 and  $B2: TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$   
 and  $B3: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel$  and  $B4: \text{preorder } Rel$   
 and  $B5: \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)}$   
 by *blast*  
 from  $B2 B4$  have  $B6: \text{refl } TRel$   
   unfolding *preorder-on-def refl-on-def*  
 by *blast*  
 from  $B2 B4$  have  $B7: \text{trans } TRel$   
   unfolding *trans-def preorder-on-def*  
 by *blast*  
 hence  $B8: TRel^+ = TRel$   
   using *trancl-id*[of  $TRel$ ]  
   by *simp*  
 with  $B6$  have  $B9: TRel^* = TRel$   
   using *reflcl-trancl*[of  $TRel$ ]  
   unfolding *refl-on-def*  
 by *blast*  
 with  $B3$  have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel^*$   
   by *simp*  
 moreover from  $B2 B8$  have  $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel$   
 and  $\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
   by *auto*  
 ultimately have  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel)$   
    $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$   
    $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
    $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([S], T) \in TRel^*)$   
    $\wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)}$   
   using  $B1 B5$   
 by *blast*  
 hence *operational-corresponding* ( $TRel^*$ )  
    $\wedge \text{weak-reduction-bisimulation } (TRel^+) \text{ Target}$   
   using *OC-iff-weak-reduction-bisimulation*[where  $TRel=TRel$ ]  
   by *simp*  
 with  $B8 B9$  have *operational-corresponding*  $TRel \wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$   
   by *simp*



**moreover from**  $B6\ B7$  **have** *preorder*  $TRel$   
   **unfolding** *preorder-on-def*  
   **by** *blast*  
**ultimately show** *operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$   
    $\wedge$  *weak-reduction-bisimulation*  $TRel$  *Target*  
   **by** *blast*  
**qed**

**lemma** (*in encoding*) *OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation*:  
**fixes**  $TRel :: ('procT \times 'procT)$  *set*  
**assumes**  $eqT$ : *equivalence*  $TRel$   
**shows** (*operational-corresponding*  $TRel \wedge$  *weak-reduction-bisimulation*  $TRel$  *Target*)  $\longleftrightarrow$   
   *weak-reduction-bisimulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
**proof** (*rule iffI, erule conjE*)  
**assume**  $oc$ : *operational-corresponding*  $TRel$  **and**  $bisimT$ : *weak-reduction-bisimulation*  $TRel$  *Target*  
**show** *weak-reduction-bisimulation* (*indRelTEQ*  $TRel$ ) (*STCal* *Source* *Target*)  
**proof** *auto*  
**fix**  $P\ Q\ P'$   
**assume**  $P \sim [\cdot]T < TRel > Q$  **and**  $P \mapsto (STCal\ Source\ Target)*\ P'$   
**thus**  $\exists Q'. Q \mapsto (STCal\ Source\ Target)*\ Q' \wedge P' \sim [\cdot]T < TRel > Q'$   
**proof** (*induct arbitrary: P'*)  
**case** (*encR S*)  
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)*\ P'$   
**from this obtain**  $S'$  **where**  $A1: S \mapsto Source* S'$  **and**  $A2: S' \in S\ P'$   
   **by** (*auto simp add: STCal-steps(1)*)  
**from**  $A1$  *oc* **obtain**  $T$  **where**  $A3: [S] \mapsto Target* T$  **and**  $A4: ([S'], T) \in TRel$   
   **by** *blast*  
**from**  $A3$  **have**  $TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)*\ (TargetTerm\ T)$   
   **by** (*simp add: STCal-steps(2)*)  
**moreover have**  $P' \sim [\cdot]T < TRel > TargetTerm\ T$   
**proof** –  
   **from**  $A2$  **have**  $P' \sim [\cdot]T < TRel > TargetTerm\ ([S'])$   
     **by** (*simp add: indRelTEQ.encR*)  
   **moreover from**  $A4$  **have**  $TargetTerm\ ([S']) \sim [\cdot]T < TRel > TargetTerm\ T$   
     **by** (*rule indRelTEQ.target*)  
   **ultimately show**  $P' \sim [\cdot]T < TRel > TargetTerm\ T$   
     **by** (*rule indRelTEQ.trans*)  
**qed**  
**ultimately show**  $\exists Q'. TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)*\ Q' \wedge P' \sim [\cdot]T < TRel > Q'$   
   **by** *blast*  
**next**  
**case** (*encL S*)  
**assume**  $TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)*\ P'$   
**from this obtain**  $T$  **where**  $B1: [S] \mapsto Target* T$  **and**  $B2: T \in T\ P'$   
   **by** (*auto simp add: STCal-steps(2)*)  
**from**  $B1$  *oc* **obtain**  $S'$  **where**  $B3: S \mapsto Source* S'$  **and**  $B4: ([S'], T) \in TRel$   
   **by** *blast*  
**from**  $B3$  **have**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)*\ (SourceTerm\ S')$   
   **by** (*simp add: STCal-steps(1)*)  
**moreover have**  $P' \sim [\cdot]T < TRel > SourceTerm\ S'$   
**proof** –  
   **from**  $B4$   $eqT$  **have**  $(T, [S']) \in TRel$   
     **unfolding** *equiv-def sym-def*  
     **by** *blast*  
   **with**  $B2$  **have**  $P' \sim [\cdot]T < TRel > TargetTerm\ ([S'])$   
     **by** (*simp add: indRelTEQ.target*)  
   **moreover have**  $TargetTerm\ ([S']) \sim [\cdot]T < TRel > SourceTerm\ S'$   
     **by** (*rule indRelTEQ.encL*)  
   **ultimately show**  $P' \sim [\cdot]T < TRel > SourceTerm\ S'$   
     **by** (*rule indRelTEQ.trans*)  
**qed**

**ultimately show**  $\exists Q'. \text{SourceTerm } S \mapsto (\text{STCal Source Target})^* Q' \wedge P' \sim [\cdot] T < \text{TRel} > Q'$   
**by blast**

**next**

**case** (*target T1 T2*)  
**assume**  $\text{TargetTerm } T1 \mapsto (\text{STCal Source Target})^* P'$   
**from this obtain**  $T1'$  **where**  $C1: T1 \mapsto \text{Target}^* T1'$  **and**  $C2: T1' \in T P'$   
**by** (*auto simp add: STCal-steps(2)*)  
**assume**  $(T1, T2) \in \text{TRel}$   
**with**  $C1$  **bisimT obtain**  $T2'$  **where**  $C3: T2 \mapsto \text{Target}^* T2'$  **and**  $C4: (T1', T2') \in \text{TRel}$   
**by blast**  
**from**  $C3$  **have**  $\text{TargetTerm } T2 \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T2')$   
**by** (*simp add: STCal-steps(2)*)  
**moreover from**  $C2 C4$  **have**  $P' \sim [\cdot] T < \text{TRel} > \text{TargetTerm } T2'$   
**by** (*simp add: indRelTEQ.target*)  
**ultimately show**  $\exists Q'. \text{TargetTerm } T2 \mapsto (\text{STCal Source Target})^* Q' \wedge P' \sim [\cdot] T < \text{TRel} > Q'$   
**by blast**

**next**

**case** (*trans P Q R*)  
**assume**  $P \mapsto (\text{STCal Source Target})^* P'$   
**and**  $\bigwedge P'. P \mapsto (\text{STCal Source Target})^* P'$   
 $\implies \exists Q'. Q \mapsto (\text{STCal Source Target})^* Q' \wedge P' \sim [\cdot] T < \text{TRel} > Q'$   
**from this obtain**  $Q'$  **where**  $D1: Q \mapsto (\text{STCal Source Target})^* Q'$  **and**  $D2: P' \sim [\cdot] T < \text{TRel} > Q'$   
**by blast**  
**assume**  $\bigwedge Q'. Q \mapsto (\text{STCal Source Target})^* Q'$   
 $\implies \exists R'. R \mapsto (\text{STCal Source Target})^* R' \wedge Q' \sim [\cdot] T < \text{TRel} > R'$   
**with**  $D1$  **obtain**  $R'$  **where**  $D3: R \mapsto (\text{STCal Source Target})^* R'$  **and**  $D4: Q' \sim [\cdot] T < \text{TRel} > R'$   
**by blast**  
**from**  $D2 D4$  **have**  $P' \sim [\cdot] T < \text{TRel} > R'$   
**by** (*rule indRelTEQ.trans*)  
**with**  $D3$  **show**  $\exists R'. R \mapsto (\text{STCal Source Target})^* R' \wedge P' \sim [\cdot] T < \text{TRel} > R'$   
**by blast**

**qed**

**next**

**fix**  $P Q Q'$   
**assume**  $P \sim [\cdot] T < \text{TRel} > Q$  **and**  $Q \mapsto (\text{STCal Source Target})^* Q'$   
**thus**  $\exists P'. P \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim [\cdot] T < \text{TRel} > Q'$   
**proof** (*induct arbitrary: Q'*)  
**case** (*encR S*)  
**assume**  $\text{TargetTerm } ([S]) \mapsto (\text{STCal Source Target})^* Q'$   
**from this obtain**  $T$  **where**  $E1: [S] \mapsto \text{Target}^* T$  **and**  $E2: T \in T Q'$   
**by** (*auto simp add: STCal-steps(2)*)  
**from**  $E1$  **oc obtain**  $S'$  **where**  $E3: S \mapsto \text{Source}^* S'$  **and**  $E4: ([S'], T) \in \text{TRel}$   
**by blast**  
**from**  $E3$  **have**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
**by** (*simp add: STCal-steps(1)*)  
**moreover have**  $\text{SourceTerm } S' \sim [\cdot] T < \text{TRel} > Q'$   
**proof** –  
**have**  $\text{SourceTerm } S' \sim [\cdot] T < \text{TRel} > \text{TargetTerm } ([S'])$   
**by** (*rule indRelTEQ.encR*)  
**moreover from**  $E2 E4$  **have**  $\text{TargetTerm } ([S']) \sim [\cdot] T < \text{TRel} > Q'$   
**by** (*simp add: indRelTEQ.target*)  
**ultimately show**  $\text{SourceTerm } S' \sim [\cdot] T < \text{TRel} > Q'$   
**by** (*rule indRelTEQ.trans*)  
**qed**

**ultimately show**  $\exists P'. \text{SourceTerm } S \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim [\cdot] T < \text{TRel} > Q'$   
**by blast**

**next**

**case** (*encL S*)  
**assume**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* Q'$   
**from this obtain**  $S'$  **where**  $F1: S \mapsto \text{Source}^* S'$  **and**  $F2: S' \in S Q'$   
**by** (*auto simp add: STCal-steps(1)*)

**from**  $F1$  **oc obtain**  $T$  **where**  $F3: \llbracket S \rrbracket \mapsto \text{Target}^* T$  **and**  $F4: (\llbracket S \rrbracket, T) \in \text{TRel}$   
**by** *blast*  
**from**  $F3$  **have**  $\text{TargetTerm } (\llbracket S \rrbracket) \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T)$   
**by** (*simp add: STCal-steps(2)*)  
**moreover have**  $\text{TargetTerm } T \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**proof** –  
**from**  $F4$  *eqT* **have**  $(T, \llbracket S \rrbracket) \in \text{TRel}$   
**unfolding** *equiv-def sym-def*  
**by** *blast*  
**hence**  $\text{TargetTerm } T \sim \llbracket \cdot \rrbracket T < \text{TRel} > \text{TargetTerm } (\llbracket S \rrbracket)$   
**by** (*rule indRelTEQ.target*)  
**moreover from**  $F2$  **have**  $\text{TargetTerm } (\llbracket S \rrbracket) \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** (*simp add: indRelTEQ.encL*)  
**ultimately show**  $\text{TargetTerm } T \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** (*rule indRelTEQ.trans*)  
**qed**  
**ultimately show**  $\exists P'. \text{TargetTerm } (\llbracket S \rrbracket) \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $\text{TargetTerm } T2 \mapsto (\text{STCal Source Target})^* Q'$   
**from this obtain**  $T2'$  **where**  $G1: T2 \mapsto \text{Target}^* T2'$  **and**  $G2: T2' \in T Q'$   
**by** (*auto simp add: STCal-steps(2)*)  
**assume**  $(T1, T2) \in \text{TRel}$   
**with**  $G1$  *bisimT* **obtain**  $T1'$  **where**  $G3: T1 \mapsto \text{Target}^* T1'$  **and**  $G4: (T1', T2') \in \text{TRel}$   
**by** *blast*  
**from**  $G3$  **have**  $\text{TargetTerm } T1 \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T1')$   
**by** (*simp add: STCal-steps(2)*)  
**moreover from**  $G2$   $G4$  **have**  $\text{TargetTerm } T1' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** (*simp add: indRelTEQ.target*)  
**ultimately show**  $\exists P'. \text{TargetTerm } T1 \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** *blast*  
**next**  
**case** (*trans P Q R'*)  
**assume**  $R \mapsto (\text{STCal Source Target})^* R'$   
**and**  $\bigwedge R'. R \mapsto (\text{STCal Source Target})^* R'$   
 $\implies \exists Q'. Q \mapsto (\text{STCal Source Target})^* Q' \wedge Q' \sim \llbracket \cdot \rrbracket T < \text{TRel} > R'$   
**from this obtain**  $Q'$  **where**  $H1: Q \mapsto (\text{STCal Source Target})^* Q'$  **and**  $H2: Q' \sim \llbracket \cdot \rrbracket T < \text{TRel} > R'$   
**by** *blast*  
**assume**  $\bigwedge Q'. Q \mapsto (\text{STCal Source Target})^* Q'$   
 $\implies \exists P'. P \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**with**  $H1$  **obtain**  $P'$  **where**  $H3: P \mapsto (\text{STCal Source Target})^* P'$  **and**  $H4: P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$   
**by** *blast*  
**from**  $H4$   $H2$  **have**  $P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > R'$   
**by** (*rule indRelTEQ.trans*)  
**with**  $H3$  **show**  $\exists P'. P \mapsto (\text{STCal Source Target})^* P' \wedge P' \sim \llbracket \cdot \rrbracket T < \text{TRel} > R'$   
**by** *blast*  
**qed**  
**qed**  
**next**  
**assume** *bisim: weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)*  
**have** *operational-corresponding TRel*  
**proof** *auto*  
**fix**  $S S'$   
**have**  $\text{SourceTerm } S \sim \llbracket \cdot \rrbracket T < \text{TRel} > \text{TargetTerm } (\llbracket S \rrbracket)$   
**by** (*rule indRelTEQ.encR*)  
**moreover assume**  $S \mapsto \text{Source}^* S'$   
**hence**  $\text{SourceTerm } S \mapsto (\text{STCal Source Target})^* (\text{SourceTerm } S')$   
**by** (*simp add: STCal-steps(1)*)  
**ultimately obtain**  $Q'$  **where**  $I1: \text{TargetTerm } (\llbracket S \rrbracket) \mapsto (\text{STCal Source Target})^* Q'$   
**and**  $I2: \text{SourceTerm } S' \sim \llbracket \cdot \rrbracket T < \text{TRel} > Q'$

```

    using bisim
  by blast
from I1 obtain T where I3:  $\llbracket S \rrbracket \mapsto \text{Target}^* T$  and I4:  $T \in T Q'$ 
  by (auto simp add: STCal-steps(2))
from eqT have  $T\text{Rel}^* = T\text{Rel}$ 
  using reflcl-trancl[of TRel] trancl-id[of TRel]
  unfolding equiv-def refl-on-def
  by auto
with I2 I4 have  $(\llbracket S \rrbracket, T) \in T\text{Rel}$ 
  using indRelTEQ-to-TRel(2)[where  $T\text{Rel}=T\text{Rel}$ ]
  trans-closure-of-TRel-refl-cond[where  $T\text{Rel}=T\text{Rel}$ ]
  by simp
with I3 show  $\exists T. \llbracket S \rrbracket \mapsto \text{Target}^* T \wedge (\llbracket S \rrbracket, T) \in T\text{Rel}$ 
  by blast
next
fix S T
have SourceTerm  $S \sim \llbracket \cdot \rrbracket T < T\text{Rel} > \text{TargetTerm } (\llbracket S \rrbracket)$ 
  by (rule indRelTEQ.encR)
moreover assume  $\llbracket S \rrbracket \mapsto \text{Target}^* T$ 
hence TargetTerm  $(\llbracket S \rrbracket) \mapsto (\text{STCal } \text{Source } \text{Target})^* (\text{TargetTerm } T)$ 
  by (simp add: STCal-steps(2))
ultimately obtain Q' where J1: SourceTerm  $S \mapsto (\text{STCal } \text{Source } \text{Target})^* Q'$ 
  and J2:  $Q' \sim \llbracket \cdot \rrbracket T < T\text{Rel} > \text{TargetTerm } T$ 
  using bisim
  by blast
from J1 obtain S' where J3:  $S \mapsto \text{Source}^* S'$  and J4:  $S' \in S Q'$ 
  by (auto simp add: STCal-steps(1))
from eqT have  $T\text{Rel}^* = T\text{Rel}$ 
  using reflcl-trancl[of TRel] trancl-id[of TRel]
  unfolding equiv-def refl-on-def
  by auto
with J2 J4 have  $(\llbracket S' \rrbracket, T) \in T\text{Rel}$ 
  using indRelTEQ-to-TRel(2)[where  $T\text{Rel}=T\text{Rel}$ ]
  trans-closure-of-TRel-refl-cond[where  $T\text{Rel}=T\text{Rel}$ ]
  by blast
with J3 show  $\exists S'. S \mapsto \text{Source}^* S' \wedge (\llbracket S' \rrbracket, T) \in T\text{Rel}$ 
  by blast
qed
moreover have weak-reduction-bisimulation TRel Target
proof -
  from eqT have  $T\text{Rel}^* = T\text{Rel}$ 
    using reflcl-trancl[of TRel] trancl-id[of TRel]
    unfolding equiv-def refl-on-def
    by auto
  with bisim show weak-reduction-bisimulation TRel Target
    using indRelTEQ-impl-TRel-is-weak-reduction-bisimulation[where  $T\text{Rel}=T\text{Rel}$ ]
    by simp
qed
ultimately show operational-corresponding TRel  $\wedge$  weak-reduction-bisimulation TRel Target
  by simp
qed

lemma (in encoding) OC-wrt-equivalence-iff-weak-reduction-bisimulation:
fixes TRel :: ('procT  $\times$  'procT) set
assumes eqT: equivalence TRel
shows (operational-corresponding TRel  $\wedge$  weak-reduction-bisimulation TRel Target)  $\longleftrightarrow$  ( $\exists \text{Rel}.$ 
 $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel} \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel})$ 
 $\wedge T\text{Rel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$ 
 $\wedge \text{trans } \text{Rel} \wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target})$ )
proof (rule iffI, erule conjE)
  assume oc: operational-corresponding TRel and bisimT: weak-reduction-bisimulation TRel Target

```

**from**  $eqT$  **have**  $rt: TRel^* = TRel$   
**using**  $reflcl-trancl[of\ TRel]$   $trancl-id[of\ TRel]$   
**unfolding**  $equiv-def$   $refl-on-def$   
**by**  $auto$   
**have**  $\forall S. SourceTerm\ S \sim[\cdot]T < TRel > TargetTerm\ (\llbracket S \rrbracket) \wedge TargetTerm\ (\llbracket S \rrbracket) \sim[\cdot]T < TRel > SourceTerm\ S$   
**by** ( $simp\ add: indRelTEQ.encR\ indRelTEQ.encL$ )  
**moreover from**  $rt$  **have**  $TRel = \{(T1, T2). TargetTerm\ T1 \sim[\cdot]T < TRel > TargetTerm\ T2\}$   
**using**  $indRelTEQ-to-TRel(4)[where\ TRel=TRel]$   
 $trans-closure-of-TRel-refl-cond[where\ TRel=TRel]$   
**by** ( $auto\ simp\ add: indRelTEQ.target$ )  
**moreover have**  $trans\ (indRelTEQ\ TRel)$   
**using**  $indRelTEQ.trans[where\ TRel=TRel]$   
**unfolding**  $trans-def$   
**by**  $blast$   
**moreover from**  $eqT$  **oc**  $bisimT$   
**have**  $weak-reduction-bisimulation\ (indRelTEQ\ TRel)\ (STCal\ Source\ Target)$   
**using**  $OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation[where\ TRel=TRel]$   
**by**  $blast$   
**ultimately**  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\} \wedge trans\ Rel$   
 $\wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$   
**by**  $blast$   
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\} \wedge trans\ Rel$   
 $\wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$   
**from**  $this$  **obtain**  $Rel$  **where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$   
**and**  $A2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$  **and**  $A3: trans\ Rel$   
**and**  $A4: weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$   
**by**  $blast$   
**have**  $operational-corresponding\ TRel$   
**proof**  $auto$   
**fix**  $S\ S'$   
**from**  $A1$  **have**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**by**  $simp$   
**moreover assume**  $S \mapsto_{Source^*} S'$   
**hence**  $SourceTerm\ S \mapsto_{(STCal\ Source\ Target)^*} (SourceTerm\ S')$   
**by** ( $simp\ add: STCal-steps(1)$ )  
**ultimately obtain**  $Q'$  **where**  $B1: TargetTerm\ (\llbracket S \rrbracket) \mapsto_{(STCal\ Source\ Target)^*} Q'$   
**and**  $B2: (SourceTerm\ S', Q') \in Rel$   
**using**  $A4$   
**by**  $blast$   
**from**  $B1$  **obtain**  $T$  **where**  $B3: \llbracket S \rrbracket \mapsto_{Target^*} T$  **and**  $B4: T \in T\ Q'$   
**by** ( $auto\ simp\ add: STCal-steps(2)$ )  
**from**  $A1$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S') \in Rel$   
**by**  $simp$   
**with**  $B2\ A3$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), Q') \in Rel$   
**unfolding**  $trans-def$   
**by**  $blast$   
**with**  $B4\ A2$  **have**  $(\llbracket S \rrbracket, T) \in TRel$   
**by**  $simp$   
**with**  $B3$  **show**  $\exists T. \llbracket S \rrbracket \mapsto_{Target^*} T \wedge (\llbracket S \rrbracket, T) \in TRel$   
**by**  $blast$   
**next**  
**fix**  $S\ T$   
**from**  $A1$  **have**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**by**  $simp$

**moreover assume**  $\llbracket S \rrbracket \mapsto \text{Target}^* T$   
**hence**  $\text{TargetTerm } (\llbracket S \rrbracket) \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } T)$   
**by** (*simp add: STCal-steps(2)*)  
**ultimately obtain**  $P'$  **where**  $C1: \text{SourceTerm } S \mapsto (\text{STCal Source Target})^* P'$   
**and**  $C2: (P', \text{TargetTerm } T) \in \text{Rel}$   
**using**  $A4$   
**by** *blast*  
**from**  $C1$  **obtain**  $S'$  **where**  $C3: S \mapsto \text{Source}^* S'$  **and**  $C4: S' \in S P'$   
**by** (*auto simp add: STCal-steps(1)*)  
**from**  $A1 C4$  **have**  $(\text{TargetTerm } (\llbracket S \rrbracket), P') \in \text{Rel}$   
**by** *simp*  
**from**  $A3$  **this**  $C2$  **have**  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}$   
**unfolding** *trans-def*  
**by** *blast*  
**with**  $A2$  **have**  $(\llbracket S \rrbracket, T) \in \text{TRel}$   
**by** *simp*  
**with**  $C3$  **show**  $\exists S'. S \mapsto \text{Source}^* S' \wedge (\llbracket S \rrbracket, T) \in \text{TRel}$   
**by** *blast*  
**qed**  
**moreover have** *weak-reduction-bisimulation TRel Target*  
**proof** *auto*  
**fix**  $TP TQ TP'$   
**assume**  $(TP, TQ) \in \text{TRel}$   
**with**  $A2$  **have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$   
**by** *simp*  
**moreover assume**  $TP \mapsto \text{Target}^* TP'$   
**hence**  $\text{TargetTerm } TP \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TP')$   
**by** (*simp add: STCal-steps(2)*)  
**ultimately obtain**  $Q'$  **where**  $D1: \text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* Q'$   
**and**  $D2: (\text{TargetTerm } TP', Q') \in \text{Rel}$   
**using**  $A4$   
**by** *blast*  
**from**  $D1$  **obtain**  $TQ'$  **where**  $D3: TQ \mapsto \text{Target}^* TQ'$  **and**  $D4: TQ' \in T Q'$   
**by** (*auto simp add: STCal-steps(2)*)  
**from**  $A2 D2 D4$  **have**  $(TP', TQ') \in \text{TRel}$   
**by** *simp*  
**with**  $D3$  **show**  $\exists TQ'. TQ \mapsto \text{Target}^* TQ' \wedge (TP', TQ') \in \text{TRel}$   
**by** *blast*  
**next**  
**fix**  $TP TQ TQ'$   
**assume**  $(TP, TQ) \in \text{TRel}$   
**with**  $A2$  **have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$   
**by** *simp*  
**moreover assume**  $TQ \mapsto \text{Target}^* TQ'$   
**hence**  $\text{TargetTerm } TQ \mapsto (\text{STCal Source Target})^* (\text{TargetTerm } TQ')$   
**by** (*simp add: STCal-steps(2)*)  
**ultimately obtain**  $P'$  **where**  $E1: \text{TargetTerm } TP \mapsto (\text{STCal Source Target})^* P'$   
**and**  $E2: (P', \text{TargetTerm } TQ') \in \text{Rel}$   
**using**  $A4$   
**by** *blast*  
**from**  $E1$  **obtain**  $TP'$  **where**  $E3: TP \mapsto \text{Target}^* TP'$  **and**  $E4: TP' \in T P'$   
**by** (*auto simp add: STCal-steps(2)*)  
**from**  $A2 E2 E4$  **have**  $(TP', TQ') \in \text{TRel}$   
**by** *simp*  
**with**  $E3$  **show**  $\exists TP'. TP \mapsto \text{Target}^* TP' \wedge (TP', TQ') \in \text{TRel}$   
**by** *blast*  
**qed**  
**ultimately show** *operational-corresponding TRel  $\wedge$  weak-reduction-bisimulation TRel Target*  
**by** *simp*  
**qed**

An encoding is strong operational corresponding w.r.t a strong bisimulation on target terms  $TRel$  iff there exists a relation, like  $indRelRTPO$ , that relates at least all source terms and their literal translations, includes  $TRel$ , and is a strong bisimulation. Thus this variant of operational correspondence ensures that source terms and their translations are strong bisimilar.

**lemma** (in *encoding*) *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation*:

**fixes**  $TRel :: ('procT \times 'procT)$  set

**shows** (strongly-operational-corresponding  $(TRel^*)$ )

$\wedge$  strong-reduction-bisimulation  $(TRel^+)$  Target

= strong-reduction-bisimulation  $(indRelRTPO\ TRel)$  (STCal Source Target)

**proof** (rule *iffI*, erule *conjE*)

**assume** *ocorr*: strongly-operational-corresponding  $(TRel^*)$

**and** *bisim*: strong-reduction-bisimulation  $(TRel^+)$  Target

**hence** strong-reduction-simulation  $(indRelRTPO\ TRel)$  (STCal Source Target)

**using** *SOC-iff-indRelRTPO-is-strong-reduction-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**moreover from** *bisim* **have** strong-reduction-simulation  $((TRel^+)^{-1})$  Target

**using** strong-reduction-bisimulations-impl-inverse-is-simulation[**where**  $Rel = TRel^+$ ]

**by** *simp*

**with** *ocorr*

**have** strong-reduction-simulation  $((indRelRTPO\ TRel)^{-1})$  (STCal Source Target)

**using** *SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**ultimately show** strong-reduction-bisimulation  $(indRelRTPO\ TRel)$  (STCal Source Target)

**using** strong-reduction-simulations-impl-bisimulation[**where**  $Rel = indRelRTPO\ TRel$ ]

**by** *simp*

**next**

**assume** *bisim*: strong-reduction-bisimulation  $(indRelRTPO\ TRel)$  (STCal Source Target)

**hence** strongly-operational-complete  $(TRel^*) \wedge$  strong-reduction-simulation  $(TRel^+)$  Target

**using** *SOC-iff-indRelRTPO-is-strong-reduction-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**moreover from** *bisim*

**have** strong-reduction-simulation  $((indRelRTPO\ TRel)^{-1})$  (STCal Source Target)

**using** strong-reduction-bisimulations-impl-inverse-is-simulation[**where**  $Rel = indRelRTPO\ TRel$ ]

**by** *simp*

**hence** strongly-operational-sound  $(TRel^*) \wedge$  strong-reduction-simulation  $((TRel^+)^{-1})$  Target

**using** *SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**ultimately show** strongly-operational-corresponding  $(TRel^*)$

$\wedge$  strong-reduction-bisimulation  $(TRel^+)$  Target

**using** strong-reduction-simulations-impl-bisimulation[**where**  $Rel = TRel^+$ ]

**by** *simp*

**qed**

**lemma** (in *encoding*) *SOC-iff-strong-reduction-bisimulation*:

**fixes**  $TRel :: ('procT \times 'procT)$  set

**shows** (strongly-operational-corresponding  $(TRel^*)$ )

$\wedge$  strong-reduction-bisimulation  $(TRel^+)$  Target

=  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$

$\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$

$\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$

$\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$

$\wedge$  strong-reduction-bisimulation  $Rel$  (STCal Source Target))

**proof** (rule *iffI*, erule *conjE*)

**have**  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelRTPO\ TRel$

**by** (*simp add: indRelRTPO.encR*)

**moreover have**  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow TargetTerm\ T1 \lesssim_{[\cdot]RT<TRel>} TargetTerm\ T2$

**by** (*simp add: indRelRTPO.target*)

**moreover have**  $\forall T1\ T2. TargetTerm\ T1 \lesssim_{[\cdot]RT<TRel>} TargetTerm\ T2 \longrightarrow (T1, T2) \in TRel^+$

**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]

**by** *simp*

**moreover have**  $\forall S T. \text{SourceTerm } S \lesssim_{[\![\ ]\!]RT < TRel >} \text{TargetTerm } T \longrightarrow ([\![S]\!] , T) \in TRel^*$   
**using** *indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond*  
**by** *simp*  
**moreover assume** *strongly-operational-corresponding (TRel\*)*  
**and** *strong-reduction-bisimulation (TRel<sup>+</sup>) Target*  
**hence** *strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)*  
**using** *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]*  
**by** *simp*  
**ultimately show**  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!] )) \in Rel)$   
 $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([\![S]\!] , T) \in TRel^*)$   
 $\wedge$  *strong-reduction-bisimulation Rel (STCal Source Target)*  
**by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!] )) \in Rel)$   
 $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([\![S]\!] , T) \in TRel^*)$   
 $\wedge$  *strong-reduction-bisimulation Rel (STCal Source Target)*  
**from this obtain Rel where**  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!] )) \in Rel$   
**and**  $A2: \forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel$   
**and**  $A3: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $A4: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([\![S]\!] , T) \in TRel^*$   
**and**  $A5: \text{strong-reduction-bisimulation Rel (STCal Source Target)}$   
**by** *blast*  
**hence** *strongly-operational-complete (TRel\*)*  
 $\wedge$  *strong-reduction-simulation (TRel<sup>+</sup>) Target*  
**using** *SOCCom-iff-strong-reduction-simulation[where TRel=TRel]*  
**by** *blast*  
**moreover from A5 have** *strong-reduction-simulation (Rel<sup>-1</sup>) (STCal Source Target)*  
**using** *strong-reduction-bisimulations-impl-inverse-is-simulation[where Rel=Rel]*  
**by** *simp*  
**with A1 A2 A3 A4 have** *strongly-operational-sound (TRel\*)*  
 $\wedge$  *strong-reduction-simulation ((TRel<sup>+</sup>)<sup>-1</sup>) Target*  
**using** *SOSou-iff-strong-reduction-simulation[where TRel=TRel]*  
**by** *blast*  
**ultimately show** *strongly-operational-corresponding (TRel\*)*  
 $\wedge$  *strong-reduction-bisimulation (TRel<sup>+</sup>) Target*  
**using** *strong-reduction-simulations-impl-bisimulation[where Rel=TRel<sup>+</sup>]*  
**by** *simp*  
**qed**

**lemma (in encoding) SOC-wrt-preorder-iff-strong-reduction-bisimulation:**

**fixes**  $TRel :: ('procT \times 'procT)$  *set*

**shows** *(strongly-operational-corresponding TRel  $\wedge$  preorder TRel*

$\wedge$  *strong-reduction-bisimulation TRel Target)*

$= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([\![S]\!] )) \in Rel)$

$\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow ([\![S]\!] , T) \in TRel)$

$\wedge$  *preorder Rel*

$\wedge$  *strong-reduction-bisimulation Rel (STCal Source Target))*

**proof** *(rule iffI, erule conjE, erule conjE, erule conjE)*

**assume**  $A1: \text{strongly-operational-complete } TRel$  **and**  $A2: \text{strongly-operational-sound } TRel$

**and**  $A3: \text{preorder } TRel$  **and**  $A4: \text{strong-reduction-bisimulation } TRel \text{ Target}$

**from**  $A3$  **have**  $A5: TRel^+ = TRel$

**using** *trancl-id[of TRel]*

**unfolding** *preorder-on-def*

**by** *blast*

**with**  $A3$  **have**  $TRel^* = TRel$

**using** *reflcl-trancl[of TRel]*



```

    unfolding preorder-on-def refl-on-def
  by blast
with A1 A2 have strongly-operational-corresponding (TRel*)
  by simp
moreover from A4 A5 have strong-reduction-bisimulation (TRel+) Target
  by simp
ultimately
have strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
  using SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]
  by blast
moreover have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } TRel$ 
  by (simp add: indRelRTPO.encR)
moreover
have TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelRTPO } TRel\}$ 
proof auto
  fix TP TQ
  assume (TP, TQ)  $\in TRel$ 
  thus TargetTerm TP  $\lesssim [\cdot]RT < TRel >$  TargetTerm TQ
    by (rule indRelRTPO.target)
next
  fix TP TQ
  assume TargetTerm TP  $\lesssim [\cdot]RT < TRel >$  TargetTerm TQ
  with A3 show (TP, TQ)  $\in TRel$ 
    using indRelRTPO-to-TRel(4)[where TRel=TRel] trancl-id[of TRel]
    unfolding preorder-on-def
    by blast
qed
moreover from A3
have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO } TRel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^+$ 
  using indRelRTPO-to-TRel(2)[where TRel=TRel] reflcl-trancl[of TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  unfolding preorder-on-def refl-on-def
  by blast
with A3 have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO } TRel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$ 
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
moreover from A3 have refl (indRelRTPO TRel)
  unfolding preorder-on-def
  by (simp add: indRelRTPO-refl)
moreover have trans (indRelRTPO TRel)
  using indRelRTPO.trans
  unfolding trans-def
  by blast
ultimately show  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$ 
   $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$ 
   $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$ 
   $\wedge \text{preorder } Rel$ 
   $\wedge \text{strong-reduction-bisimulation } Rel (\text{STCal } \text{Source } \text{Target})$ 
  unfolding preorder-on-def
  by blast
next
assume  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$ 
   $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$ 
   $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$ 
   $\wedge \text{preorder } Rel$ 
   $\wedge \text{strong-reduction-bisimulation } Rel (\text{STCal } \text{Source } \text{Target})$ 
from this obtain Rel where B1:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel$ 
and B2:  $TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$ 
and B3:  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$ 
and B4: preorder Rel
and B5: strong-reduction-bisimulation Rel (STCal Source Target)

```

by *blast*  
 from  $B2\ B4$  have  $B6: \text{refl } T\text{Rel}$   
   **unfolding** *preorder-on-def refl-on-def*  
   by *blast*  
 from  $B2\ B4$  have  $B7: \text{trans } T\text{Rel}$   
   **unfolding** *trans-def preorder-on-def*  
   by *blast*  
 hence  $B8: T\text{Rel}^+ = T\text{Rel}$   
   by (*rule trancl-id*)  
 with  $B6$  have  $B9: T\text{Rel}^* = T\text{Rel}$   
   **using** *reflcl-trancl[of TRel]*  
   **unfolding** *refl-on-def*  
   by *blast*  
 with  $B3$  have  $\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$   
   by *simp*  
 moreover from  $B2\ B8$  have  $\forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$   
 and  $\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+$   
   by *auto*  
 ultimately have  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$   
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$   
 $\wedge \text{strong-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target})$   
   **using**  $B1\ B5$   
   by *blast*  
 hence *strongly-operational-corresponding*  $(T\text{Rel}^*) \wedge \text{strong-reduction-bisimulation } (T\text{Rel}^+) \text{Target}$   
   **using** *SOC-iff-strong-reduction-bisimulation[where TRel=TRel]*  
   by *simp*  
 with  $B8\ B9$   
 have *strongly-operational-corresponding*  $T\text{Rel} \wedge \text{strong-reduction-bisimulation } T\text{Rel } \text{Target}$   
   by *simp*  
 moreover from  $B6\ B7$  have *preorder*  $T\text{Rel}$   
   **unfolding** *preorder-on-def*  
   by *blast*  
 ultimately **show** *strongly-operational-corresponding*  $T\text{Rel} \wedge \text{preorder } T\text{Rel}$   
    $\wedge \text{strong-reduction-bisimulation } T\text{Rel } \text{Target}$   
   by *blast*  
 qed

**lemma** (*in encoding*) *SOC-wrt-TRel-iff-strong-reduction-bisimulation*:

**shows**  $(\exists T\text{Rel}. \text{strongly-operational-corresponding } (T\text{Rel}^*)$   
 $\wedge \text{strong-reduction-bisimulation } (T\text{Rel}^+) \text{Target})$   
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}$   
 $\longrightarrow (\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}^=)$   
 $\wedge \text{strong-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target}))$

**proof** (*rule iffI*)

**assume**  $\exists T\text{Rel}. \text{strongly-operational-corresponding } (T\text{Rel}^*)$   
 $\wedge \text{strong-reduction-bisimulation } (T\text{Rel}^+) \text{Target}$   
**from this obtain**  $T\text{Rel}$  **where** *strongly-operational-corresponding*  $(T\text{Rel}^*)$   
   **and** *strong-reduction-bisimulation*  $(T\text{Rel}^+) \text{Target}$   
   by *blast*  
**hence** *strong-reduction-bisimulation*  $(\text{indRelRTPO } T\text{Rel}) (\text{STCal } \text{Source } \text{Target})$   
   **using** *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]*  
   by *simp*  
**moreover have**  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{indRelRTPO } T\text{Rel}$   
   by (*simp add: indRelRTPO.encR*)  
**moreover have**  $\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in (\text{indRelRTPO } T\text{Rel})$   
 $\longrightarrow (\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in (\text{indRelRTPO } T\text{Rel})^=$   
   **using** *indRelRTPO-relates-source-target[where TRel=TRel]*  
   by *simp*

**ultimately show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (TargetTerm\ (\llbracket S \rrbracket), TargetTerm\ T) \in Rel^=)$   
 $\wedge$  *strong-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)  
**by** *blast*

**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (TargetTerm\ (\llbracket S \rrbracket), TargetTerm\ T) \in Rel^=)$   
 $\wedge$  *strong-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)  
**from this obtain** *Rel* **where** *A1*:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and** *A2*:  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel$   
 $\longrightarrow (TargetTerm\ (\llbracket S \rrbracket), TargetTerm\ T) \in Rel^=$   
**and** *A3*: *strong-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)  
**by** *blast*

**from** *A2* **obtain** *TRel* **where**  $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**using** *target-relation-from-source-target-relation*[**where** *Rel*=*Rel*]  
**by** *blast*

**with** *A1* *A3* **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge$  *strong-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)  
**by** *blast*

**hence** *strongly-operational-corresponding* (*TRel*<sup>\*</sup>)  
 $\wedge$  *strong-reduction-bisimulation* (*TRel*<sup>+</sup>) *Target*  
**using** *SOC-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]  
**by** *simp*

**thus**  $\exists TRel. \text{strongly-operational-corresponding } (TRel^*)$   
 $\wedge$  *strong-reduction-bisimulation* (*TRel*<sup>+</sup>) *Target*  
**by** *blast*

**qed**

**lemma** (**in** *encoding*) *SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation*:  
**fixes** *TRel* :: ('*procT* × '*procT*) *set*  
**assumes** *eqT*: *equivalence* *TRel*  
**shows** (*strongly-operational-corresponding* *TRel*  $\wedge$  *strong-reduction-bisimulation* *TRel* *Target*)  
 $\longleftrightarrow$  *strong-reduction-bisimulation* (*indRelTEQ* *TRel*) (*STCal* *Source* *Target*)  
**proof** (*rule* *iffI*, *erule* *conjE*)  
**assume** *oc*: *strongly-operational-corresponding* *TRel*  
**and** *bisimT*: *strong-reduction-bisimulation* *TRel* *Target*  
**show** *strong-reduction-bisimulation* (*indRelTEQ* *TRel*) (*STCal* *Source* *Target*)  
**proof** *auto*  
**fix** *P* *Q* *P'*  
**assume**  $P \sim \llbracket \cdot \rrbracket T < TRel > Q$  **and**  $P \mapsto (STCal\ Source\ Target)\ P'$   
**thus**  $\exists Q'. Q \mapsto (STCal\ Source\ Target)\ Q' \wedge P' \sim \llbracket \cdot \rrbracket T < TRel > Q'$   
**proof** (*induct* *arbitrary*: *P'*)  
**case** (*encR* *S*)  
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)\ P'$   
**from this obtain** *S'* **where** *A1*:  $S \mapsto Source\ S'$  **and** *A2*:  $S' \in S\ P'$   
**by** (*auto* *simp* *add*: *STCal-step*(1))  
**from** *A1* *oc* **obtain** *T* **where** *A3*:  $\llbracket S \rrbracket \mapsto Target\ T$  **and** *A4*:  $(\llbracket S \rrbracket, T) \in TRel$   
**by** *blast*  
**from** *A3* **have**  $TargetTerm\ (\llbracket S \rrbracket) \mapsto (STCal\ Source\ Target)\ (TargetTerm\ T)$   
**by** (*simp* *add*: *STCal-step*(2))  
**moreover** **have**  $P' \sim \llbracket \cdot \rrbracket T < TRel > TargetTerm\ T$   
**proof** –  
**from** *A2* **have**  $P' \sim \llbracket \cdot \rrbracket T < TRel > TargetTerm\ (\llbracket S \rrbracket)$   
**by** (*simp* *add*: *indRelTEQ.encR*)  
**moreover** **from** *A4* **have**  $TargetTerm\ (\llbracket S \rrbracket) \sim \llbracket \cdot \rrbracket T < TRel > TargetTerm\ T$   
**by** (*rule* *indRelTEQ.target*)

ultimately show  $P' \sim [\cdot] T < TRel > TargetTerm T$   
 by (rule *indRelTEQ.trans*)  
 qed  
 ultimately show  $\exists Q'. TargetTerm ([S]) \mapsto (STCal Source Target) Q' \wedge P' \sim [\cdot] T < TRel > Q'$   
 by blast  
 next  
 case (*encL S*)  
 assume  $TargetTerm ([S]) \mapsto (STCal Source Target) P'$   
 from this obtain  $T$  where  $B1: [S] \mapsto Target T$  and  $B2: T \in T P'$   
 by (*auto simp add: STCal-step(2)*)  
 from  $B1$  oc obtain  $S'$  where  $B3: S \mapsto Source S'$  and  $B4: ([S'], T) \in TRel$   
 by blast  
 from  $B3$  have  $SourceTerm S \mapsto (STCal Source Target) (SourceTerm S')$   
 by (*simp add: STCal-step(1)*)  
 moreover have  $P' \sim [\cdot] T < TRel > SourceTerm S'$   
 proof –  
 from  $B4$  eqT have  $(T, [S']) \in TRel$   
 unfolding *equiv-def sym-def*  
 by blast  
 with  $B2$  have  $P' \sim [\cdot] T < TRel > TargetTerm ([S'])$   
 by (*simp add: indRelTEQ.target*)  
 moreover have  $TargetTerm ([S']) \sim [\cdot] T < TRel > SourceTerm S'$   
 by (*rule indRelTEQ.encL*)  
 ultimately show  $P' \sim [\cdot] T < TRel > SourceTerm S'$   
 by (*rule indRelTEQ.trans*)  
 qed  
 ultimately show  $\exists Q'. SourceTerm S \mapsto (STCal Source Target) Q' \wedge P' \sim [\cdot] T < TRel > Q'$   
 by blast  
 next  
 case (*target T1 T2*)  
 assume  $TargetTerm T1 \mapsto (STCal Source Target) P'$   
 from this obtain  $T1'$  where  $C1: T1 \mapsto Target T1'$  and  $C2: T1' \in T P'$   
 by (*auto simp add: STCal-step(2)*)  
 assume  $(T1, T2) \in TRel$   
 with  $C1$  *bisimT* obtain  $T2'$  where  $C3: T2 \mapsto Target T2'$  and  $C4: (T1', T2') \in TRel$   
 by blast  
 from  $C3$  have  $TargetTerm T2 \mapsto (STCal Source Target) (TargetTerm T2')$   
 by (*simp add: STCal-step(2)*)  
 moreover from  $C2 C4$  have  $P' \sim [\cdot] T < TRel > TargetTerm T2'$   
 by (*simp add: indRelTEQ.target*)  
 ultimately show  $\exists Q'. TargetTerm T2 \mapsto (STCal Source Target) Q' \wedge P' \sim [\cdot] T < TRel > Q'$   
 by blast  
 next  
 case (*trans P Q R*)  
 assume  $P \mapsto (STCal Source Target) P'$   
 and  $\bigwedge P'. P \mapsto (STCal Source Target) P'$   
 $\implies \exists Q'. Q \mapsto (STCal Source Target) Q' \wedge P' \sim [\cdot] T < TRel > Q'$   
 from this obtain  $Q'$  where  $D1: Q \mapsto (STCal Source Target) Q'$  and  $D2: P' \sim [\cdot] T < TRel > Q'$   
 by blast  
 assume  $\bigwedge Q'. Q \mapsto (STCal Source Target) Q'$   
 $\implies \exists R'. R \mapsto (STCal Source Target) R' \wedge Q' \sim [\cdot] T < TRel > R'$   
 with  $D1$  obtain  $R'$  where  $D3: R \mapsto (STCal Source Target) R'$  and  $D4: Q' \sim [\cdot] T < TRel > R'$   
 by blast  
 from  $D2 D4$  have  $P' \sim [\cdot] T < TRel > R'$   
 by (*rule indRelTEQ.trans*)  
 with  $D3$  show  $\exists R'. R \mapsto (STCal Source Target) R' \wedge P' \sim [\cdot] T < TRel > R'$   
 by blast  
 qed  
 next  
 fix  $P Q Q'$   
 assume  $P \sim [\cdot] T < TRel > Q$  and  $Q \mapsto (STCal Source Target) Q'$

**thus**  $\exists P'. P \mapsto (STCal\ Source\ Target) P' \wedge P' \sim[\cdot] T < TRel > Q'$   
**proof** (*induct arbitrary: Q'*)  
**case** (*encR S*)  
**assume**  $TargetTerm ([S]) \mapsto (STCal\ Source\ Target) Q'$   
**from this obtain T where**  $E1: [S] \mapsto Target\ T$  **and**  $E2: T \in T\ Q'$   
**by** (*auto simp add: STCal-step(2)*)  
**from E1 oc obtain S' where**  $E3: S \mapsto Source\ S'$  **and**  $E4: ([S'], T) \in TRel$   
**by blast**  
**from E3 have**  $SourceTerm\ S \mapsto (STCal\ Source\ Target) (SourceTerm\ S')$   
**by** (*simp add: STCal-step(1)*)  
**moreover have**  $SourceTerm\ S' \sim[\cdot] T < TRel > Q'$   
**proof** –  
**have**  $SourceTerm\ S' \sim[\cdot] T < TRel > TargetTerm ([S'])$   
**by** (*rule indRelTEQ.encR*)  
**moreover from E2 E4 have**  $TargetTerm ([S']) \sim[\cdot] T < TRel > Q'$   
**by** (*simp add: indRelTEQ.target*)  
**ultimately show**  $SourceTerm\ S' \sim[\cdot] T < TRel > Q'$   
**by** (*rule indRelTEQ.trans*)  
**qed**  
**ultimately show**  $\exists P'. SourceTerm\ S \mapsto (STCal\ Source\ Target) P' \wedge P' \sim[\cdot] T < TRel > Q'$   
**by blast**  
**next**  
**case** (*encL S*)  
**assume**  $SourceTerm\ S \mapsto (STCal\ Source\ Target) Q'$   
**from this obtain S' where**  $F1: S \mapsto Source\ S'$  **and**  $F2: S' \in S\ Q'$   
**by** (*auto simp add: STCal-step(1)*)  
**from F1 oc obtain T where**  $F3: [S] \mapsto Target\ T$  **and**  $F4: ([S'], T) \in TRel$   
**by blast**  
**from F3 have**  $TargetTerm ([S]) \mapsto (STCal\ Source\ Target) (TargetTerm\ T)$   
**by** (*simp add: STCal-step(2)*)  
**moreover have**  $TargetTerm\ T \sim[\cdot] T < TRel > Q'$   
**proof** –  
**from F4 eqT have**  $(T, [S']) \in TRel$   
**unfolding equiv-def sym-def**  
**by blast**  
**hence**  $TargetTerm\ T \sim[\cdot] T < TRel > TargetTerm ([S'])$   
**by** (*rule indRelTEQ.target*)  
**moreover from F2 have**  $TargetTerm ([S']) \sim[\cdot] T < TRel > Q'$   
**by** (*simp add: indRelTEQ.encL*)  
**ultimately show**  $TargetTerm\ T \sim[\cdot] T < TRel > Q'$   
**by** (*rule indRelTEQ.trans*)  
**qed**  
**ultimately show**  $\exists P'. TargetTerm ([S]) \mapsto (STCal\ Source\ Target) P' \wedge P' \sim[\cdot] T < TRel > Q'$   
**by blast**  
**next**  
**case** (*target T1 T2*)  
**assume**  $TargetTerm\ T2 \mapsto (STCal\ Source\ Target) Q'$   
**from this obtain T2' where**  $G1: T2 \mapsto Target\ T2'$  **and**  $G2: T2' \in T\ Q'$   
**by** (*auto simp add: STCal-step(2)*)  
**assume**  $(T1, T2) \in TRel$   
**with G1 bisimT obtain T1' where**  $G3: T1 \mapsto Target\ T1'$  **and**  $G4: (T1', T2') \in TRel$   
**by blast**  
**from G3 have**  $TargetTerm\ T1 \mapsto (STCal\ Source\ Target) (TargetTerm\ T1')$   
**by** (*simp add: STCal-step(2)*)  
**moreover from G2 G4 have**  $TargetTerm\ T1' \sim[\cdot] T < TRel > Q'$   
**by** (*simp add: indRelTEQ.target*)  
**ultimately show**  $\exists P'. TargetTerm\ T1 \mapsto (STCal\ Source\ Target) P' \wedge P' \sim[\cdot] T < TRel > Q'$   
**by blast**  
**next**  
**case** (*trans P Q R R'*)  
**assume**  $R \mapsto (STCal\ Source\ Target) R'$

**and**  $\bigwedge R'. R \mapsto (STCal\ Source\ Target)\ R'$   
 $\implies \exists Q'. Q \mapsto (STCal\ Source\ Target)\ Q' \wedge Q' \sim [\cdot] T < TRel > R'$   
**from this obtain**  $Q'$  **where**  $H1: Q \mapsto (STCal\ Source\ Target)\ Q'$  **and**  $H2: Q' \sim [\cdot] T < TRel > R'$   
**by blast**  
**assume**  $\bigwedge Q'. Q \mapsto (STCal\ Source\ Target)\ Q'$   
 $\implies \exists P'. P \mapsto (STCal\ Source\ Target)\ P' \wedge P' \sim [\cdot] T < TRel > Q'$   
**with**  $H1$  **obtain**  $P'$  **where**  $H3: P \mapsto (STCal\ Source\ Target)\ P'$  **and**  $H4: P' \sim [\cdot] T < TRel > Q'$   
**by blast**  
**from**  $H4$   $H2$  **have**  $P' \sim [\cdot] T < TRel > R'$   
**by** (rule *indRelTEQ.trans*)  
**with**  $H3$  **show**  $\exists P'. P \mapsto (STCal\ Source\ Target)\ P' \wedge P' \sim [\cdot] T < TRel > R'$   
**by blast**  
**qed**  
**qed**  
**next**  
**assume** *bisim: strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)*  
**have** *strongly-operational-corresponding TRel*  
**proof auto**  
**fix**  $S\ S'$   
**have**  $SourceTerm\ S \sim [\cdot] T < TRel > TargetTerm\ ([S])$   
**by** (rule *indRelTEQ.encR*)  
**moreover assume**  $S \mapsto Source\ S'$   
**hence**  $SourceTerm\ S \mapsto (STCal\ Source\ Target)\ (SourceTerm\ S')$   
**by** (*simp add: STCal-step(1)*)  
**ultimately obtain**  $Q'$  **where**  $I1: TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)\ Q'$   
**and**  $I2: SourceTerm\ S' \sim [\cdot] T < TRel > Q'$   
**using** *bisim*  
**by blast**  
**from**  $I1$  **obtain**  $T$  **where**  $I3: [S] \mapsto Target\ T$  **and**  $I4: T \in T\ Q'$   
**by** (*auto simp add: STCal-step(2)*)  
**from** *eqT* **have**  $TRel^* = TRel$   
**using** *reflcl-trancl[of TRel] trancl-id[of TRel]*  
**unfolding** *equiv-def refl-on-def*  
**by auto**  
**with**  $I2\ I4$  **have**  $([S], T) \in TRel$   
**using** *indRelTEQ-to-TRel(2)[where TRel=TRel]*  
*trans-closure-of-TRel-refl-cond[where TRel=TRel]*  
**by simp**  
**with**  $I3$  **show**  $\exists T. [S] \mapsto Target\ T \wedge ([S], T) \in TRel$   
**by blast**  
**next**  
**fix**  $S\ T$   
**have**  $SourceTerm\ S \sim [\cdot] T < TRel > TargetTerm\ ([S])$   
**by** (rule *indRelTEQ.encR*)  
**moreover assume**  $[S] \mapsto Target\ T$   
**hence**  $TargetTerm\ ([S]) \mapsto (STCal\ Source\ Target)\ (TargetTerm\ T)$   
**by** (*simp add: STCal-step(2)*)  
**ultimately obtain**  $Q'$  **where**  $J1: SourceTerm\ S \mapsto (STCal\ Source\ Target)\ Q'$   
**and**  $J2: Q' \sim [\cdot] T < TRel > TargetTerm\ T$   
**using** *bisim*  
**by blast**  
**from**  $J1$  **obtain**  $S'$  **where**  $J3: S \mapsto Source\ S'$  **and**  $J4: S' \in S\ Q'$   
**by** (*auto simp add: STCal-step(1)*)  
**from** *eqT* **have**  $TRel^* = TRel$   
**using** *reflcl-trancl[of TRel] trancl-id[of TRel]*  
**unfolding** *equiv-def refl-on-def*  
**by auto**  
**with**  $J2\ J4$  **have**  $([S], T) \in TRel$   
**using** *indRelTEQ-to-TRel(2)[where TRel=TRel]*  
*trans-closure-of-TRel-refl-cond[where TRel=TRel]*  
**by blast**

```

with  $J\beta$  show  $\exists S'. S \mapsto \text{Source } S' \wedge (\llbracket S \rrbracket, T) \in TRel$ 
  by blast
qed
moreover have strong-reduction-bisimulation  $TRel$   $Target$ 
proof -
  from eqT have  $TRel^* = TRel$ 
    using reflcl-trancl[of  $TRel$ ] trancl-id[of  $TRel$ ]
    unfolding equiv-def refl-on-def
  by auto
  with bisim show strong-reduction-bisimulation  $TRel$   $Target$ 
    using indRelTEQ-impl-TRel-is-strong-reduction-bisimulation[where  $TRel = TRel$ ]
  by simp
qed
ultimately
show strongly-operational-corresponding  $TRel \wedge$  strong-reduction-bisimulation  $TRel$   $Target$ 
  by simp
qed

lemma (in encoding) SOC-wrt-equivalence-iff-strong-reduction-bisimulation:
fixes  $TRel :: ('procT \times 'procT) \text{ set}$ 
assumes eqT: equivalence  $TRel$ 
shows (strongly-operational-corresponding  $TRel \wedge$  strong-reduction-bisimulation  $TRel$   $Target$ )
   $\longleftrightarrow (\exists Rel.$ 
     $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in Rel)$ 
     $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$ 
     $\wedge \text{trans } Rel \wedge \text{strong-reduction-bisimulation } Rel \text{ (STCal } Source \text{ Target)})$ 
proof (rule iffI, erule conjE)
  assume oc: strongly-operational-corresponding  $TRel$ 
  and bisimT: strong-reduction-bisimulation  $TRel$   $Target$ 
  from eqT have rt:  $TRel^* = TRel$ 
    using reflcl-trancl[of  $TRel$ ] trancl-id[of  $TRel$ ]
    unfolding equiv-def refl-on-def
  by auto
  have  $\forall S. \text{SourceTerm } S \sim[\llbracket \cdot \rrbracket] T < TRel > \text{TargetTerm } (\llbracket S \rrbracket) \wedge \text{TargetTerm } (\llbracket S \rrbracket) \sim[\llbracket \cdot \rrbracket] T < TRel > \text{SourceTerm } S$ 
  by (simp add: indRelTEQ.encR indRelTEQ.encL)
  moreover from rt have  $TRel = \{(T1, T2). \text{TargetTerm } T1 \sim[\llbracket \cdot \rrbracket] T < TRel > \text{TargetTerm } T2\}$ 
    using indRelTEQ-to-TRel(4)[where  $TRel = TRel$ ]
    trans-closure-of-TRel-refl-cond[where  $TRel = TRel$ ]
  by (auto simp add: indRelTEQ.target)
  moreover have trans (indRelTEQ  $TRel$ )
    using indRelTEQ.trans[where  $TRel = TRel$ ]
    unfolding trans-def
  by blast
  moreover from eqT oc bisimT
  have strong-reduction-bisimulation (indRelTEQ  $TRel$ ) (STCal  $Source$   $Target$ )
    using SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation[where  $TRel = TRel$ ]
  by blast
  ultimately
  show  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in Rel)$ 
     $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\} \wedge \text{trans } Rel$ 
     $\wedge \text{strong-reduction-bisimulation } Rel \text{ (STCal } Source \text{ Target)}$ 
  by blast
next
  assume  $\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel$ 
     $\wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in Rel)$ 
     $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\} \wedge \text{trans } Rel$ 
     $\wedge \text{strong-reduction-bisimulation } Rel \text{ (STCal } Source \text{ Target)}$ 
  from this obtain  $Rel$  where  $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel$ 
     $\wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in Rel$ 
  and  $A2: TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$  and  $A3: \text{trans } Rel$ 

```

**and**  $A_4$ : *strong-reduction-bisimulation Rel* ( $STCal$   $Source$   $Target$ )  
**by** *blast*  
**have** *strongly-operational-corresponding TRel*  
**proof** *auto*  
**fix**  $S S'$   
**from**  $A1$  **have** ( $SourceTerm S, TargetTerm (\llbracket S \rrbracket) \in Rel$ )  
**by** *simp*  
**moreover** **assume**  $S \mapsto Source S'$   
**hence**  $SourceTerm S \mapsto (STCal Source Target) (SourceTerm S')$   
**by** (*simp add: STCal-step(1)*)  
**ultimately** **obtain**  $Q'$  **where**  $B1: TargetTerm (\llbracket S \rrbracket) \mapsto (STCal Source Target) Q'$   
**and**  $B2: (SourceTerm S', Q') \in Rel$   
**using**  $A_4$   
**by** *blast*  
**from**  $B1$  **obtain**  $T$  **where**  $B3: \llbracket S \rrbracket \mapsto Target T$  **and**  $B_4: T \in T Q'$   
**by** (*auto simp add: STCal-step(2)*)  
**from**  $A1$  **have** ( $TargetTerm (\llbracket S \rrbracket), SourceTerm S' \in Rel$ )  
**by** *simp*  
**with**  $B2 A3$  **have** ( $TargetTerm (\llbracket S \rrbracket), Q' \in Rel$ )  
**unfolding** *trans-def*  
**by** *blast*  
**with**  $B_4 A2$  **have** ( $\llbracket S \rrbracket, T \in TRel$ )  
**by** *simp*  
**with**  $B3$  **show**  $\exists T. \llbracket S \rrbracket \mapsto Target T \wedge (\llbracket S \rrbracket, T) \in TRel$   
**by** *blast*  
**next**  
**fix**  $S T$   
**from**  $A1$  **have** ( $SourceTerm S, TargetTerm (\llbracket S \rrbracket) \in Rel$ )  
**by** *simp*  
**moreover** **assume**  $\llbracket S \rrbracket \mapsto Target T$   
**hence**  $TargetTerm (\llbracket S \rrbracket) \mapsto (STCal Source Target) (TargetTerm T)$   
**by** (*simp add: STCal-step(2)*)  
**ultimately** **obtain**  $P'$  **where**  $C1: SourceTerm S \mapsto (STCal Source Target) P'$   
**and**  $C2: (P', TargetTerm T) \in Rel$   
**using**  $A_4$   
**by** *blast*  
**from**  $C1$  **obtain**  $S'$  **where**  $C3: S \mapsto Source S'$  **and**  $C_4: S' \in S P'$   
**by** (*auto simp add: STCal-step(1)*)  
**from**  $A1 C_4$  **have** ( $TargetTerm (\llbracket S \rrbracket), P' \in Rel$ )  
**by** *simp*  
**from**  $A3$  **this**  $C2$  **have** ( $TargetTerm (\llbracket S \rrbracket), TargetTerm T \in Rel$ )  
**unfolding** *trans-def*  
**by** *blast*  
**with**  $A2$  **have** ( $\llbracket S \rrbracket, T \in TRel$ )  
**by** *simp*  
**with**  $C3$  **show**  $\exists S'. S \mapsto Source S' \wedge (\llbracket S \rrbracket, T) \in TRel$   
**by** *blast*  
**qed**  
**moreover** **have** *strong-reduction-bisimulation TRel Target*  
**proof** *auto*  
**fix**  $TP TQ TP'$   
**assume** ( $TP, TQ \in TRel$ )  
**with**  $A2$  **have** ( $TargetTerm TP, TargetTerm TQ \in Rel$ )  
**by** *simp*  
**moreover** **assume**  $TP \mapsto Target TP'$   
**hence**  $TargetTerm TP \mapsto (STCal Source Target) (TargetTerm TP')$   
**by** (*simp add: STCal-step(2)*)  
**ultimately** **obtain**  $Q'$  **where**  $D1: TargetTerm TQ \mapsto (STCal Source Target) Q'$   
**and**  $D2: (TargetTerm TP', Q') \in Rel$   
**using**  $A_4$   
**by** *blast*



```

from  $D1$  obtain  $TQ'$  where  $D3: TQ \mapsto \text{Target } TQ'$  and  $D4: TQ' \in T Q'$ 
  by (auto simp add: STCal-step(2))
from  $A2 D2 D4$  have  $(TP', TQ') \in TRel$ 
  by simp
with  $D3$  show  $\exists TQ'. TQ \mapsto \text{Target } TQ' \wedge (TP', TQ') \in TRel$ 
  by blast
next
fix  $TP TQ TQ'$ 
assume  $(TP, TQ) \in TRel$ 
with  $A2$  have  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in Rel$ 
  by simp
moreover assume  $TQ \mapsto \text{Target } TQ'$ 
hence  $\text{TargetTerm } TQ \mapsto (\text{STCal Source Target}) (\text{TargetTerm } TQ')$ 
  by (simp add: STCal-step(2))
ultimately obtain  $P'$  where  $E1: \text{TargetTerm } TP \mapsto (\text{STCal Source Target}) P'$ 
  and  $E2: (P', \text{TargetTerm } TQ') \in Rel$ 
  using  $A4$ 
  by blast
from  $E1$  obtain  $TP'$  where  $E3: TP \mapsto \text{Target } TP'$  and  $E4: TP' \in T P'$ 
  by (auto simp add: STCal-step(2))
from  $A2 E2 E4$  have  $(TP', TQ') \in TRel$ 
  by simp
with  $E3$  show  $\exists TP'. TP \mapsto \text{Target } TP' \wedge (TP', TQ') \in TRel$ 
  by blast
qed
ultimately
show strongly-operational-corresponding  $TRel \wedge$  strong-reduction-bisimulation  $TRel \text{ Target}$ 
  by simp
qed

end
theory FullAbstraction
  imports SourceTargetRelation
begin

```

## 9 Full Abstraction

An encoding is fully abstract w.r.t. some source term relation  $SRel$  and some target term relation  $TRel$  if two source terms  $S1$  and  $S2$  form a pair  $(S1, S2)$  in  $SRel$  iff their literal translations form a pair  $(\text{enc } S1, \text{enc } S2)$  in  $TRel$ .

**abbreviation** (*in encoding*) *fully-abstract*  
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set} \Rightarrow \text{bool}$   
**where**  
*fully-abstract*  $SRel TRel \equiv \forall S1 S2. (S1, S2) \in SRel \iff (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$

### 9.1 Trivial Full Abstraction Results

We start with some trivial full abstraction results. Each injective encoding is fully abstract w.r.t. to the identity relation on the source and the identity relation on the target.

**lemma** (*in encoding*) *inj-enc-is-fully-abstract-wrt-identities*:  
**assumes** *injectivity*:  $\forall S1 S2. \llbracket S1 \rrbracket = \llbracket S2 \rrbracket \longrightarrow S1 = S2$   
**shows** *fully-abstract*  $\{(S1, S2). S1 = S2\} \{(T1, T2). T1 = T2\}$   
**by** (*auto simp add: injectivity*)

Each encoding is fully abstract w.r.t. the empty relation on the source and the target.

**lemma** (*in encoding*) *fully-abstract-wrt-empty-relation*:  
**shows** *fully-abstract*  $\{\} \{\}$   
**by** *auto*

Similarly, each encoding is fully abstract w.r.t. the all-relation on the source and the target.

**lemma** (in *encoding*) *fully-abstract-wrt-all-relation*:  
**shows** *fully-abstract*  $\{(S1, S2). True\} \{(T1, T2). True\}$   
**by** *auto*

If the encoding is injective then for each source term relation RelS there exists a target term relation RelT such that the encoding is fully abstract w.r.t. RelS and RelT.

**lemma** (in *encoding*) *fully-abstract-wrt-source-relation*:  
**fixes** *RelS* :: ('procS × 'procS) set  
**assumes** *injectivity*:  $\forall S1 S2. \llbracket S1 \rrbracket = \llbracket S2 \rrbracket \longrightarrow S1 = S2$   
**shows**  $\exists RelT. \text{fully-abstract } RelS \ RelT$

**proof** –  
**define** *RelT* **where**  $RelT = \{(T1, T2). \exists S1 S2. (S1, S2) \in RelS \wedge T1 = \llbracket S1 \rrbracket \wedge T2 = \llbracket S2 \rrbracket\}$   
**with** *injectivity* **have** *fully-abstract RelS RelT*  
**by** *blast*  
**thus**  $\exists RelT. \text{fully-abstract } RelS \ RelT$   
**by** *blast*  
**qed**

If all source terms that are translated to the same target term are related by a trans source term relation RelS, then there exists a target term relation RelT such that the encoding is fully abstract w.r.t. RelS and RelT.

**lemma** (in *encoding*) *fully-abstract-wrt-trans-source-relation*:  
**fixes** *RelS* :: ('procS × 'procS) set  
**assumes** *encRelS*:  $\forall S1 S2. \llbracket S1 \rrbracket = \llbracket S2 \rrbracket \longrightarrow (S1, S2) \in RelS$   
**and** *transS*: *trans RelS*  
**shows**  $\exists RelT. \text{fully-abstract } RelS \ RelT$

**proof** –  
**define** *RelT* **where**  $RelT = \{(T1, T2). \exists S1 S2. (S1, S2) \in RelS \wedge T1 = \llbracket S1 \rrbracket \wedge T2 = \llbracket S2 \rrbracket\}$   
**have** *fully-abstract RelS RelT*  
**proof** *auto*  
**fix** *S1 S2*  
**assume**  $(S1, S2) \in RelS$   
**with** *RelT-def* **show**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in RelT$   
**by** *blast*  
**next**  
**fix** *S1 S2*  
**assume**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in RelT$   
**with** *RelT-def* **obtain** *S1' S2'* **where** *A1*:  $(S1', S2') \in RelS$  **and** *A2*:  $\llbracket S1 \rrbracket = \llbracket S1' \rrbracket$   
**and** *A3*:  $\llbracket S2 \rrbracket = \llbracket S2' \rrbracket$   
**by** *blast*  
**from** *A2 encRelS* **have**  $(S1, S1') \in RelS$   
**by** *simp*  
**from** *this A1 transS* **have**  $(S1, S2') \in RelS$   
**unfolding** *trans-def*  
**by** *blast*  
**moreover** **from** *A3 encRelS* **have**  $(S2', S2) \in RelS$   
**by** *simp*  
**ultimately** **show**  $(S1, S2) \in RelS$   
**using** *transS*  
**unfolding** *trans-def*  
**by** *blast*  
**qed**  
**thus**  $\exists RelT. \text{fully-abstract } RelS \ RelT$   
**by** *blast*  
**qed**

**lemma** (in *encoding*) *fully-abstract-wrt-trans-closure-of-source-relation*:  
**fixes** *RelS* :: ('procS × 'procS) set

**assumes**  $encRelS: \forall S1\ S2. \llbracket S1 \rrbracket = \llbracket S2 \rrbracket \longrightarrow (S1, S2) \in RelS^+$   
**shows**  $\exists RelT. \text{fully-abstract } (RelS^+) RelT$   
**using**  $encRelS \text{ trans-trancl[of } RelS]$   
 $\text{fully-abstract-wrt-trans-source-relation[where } RelS=RelS^+]$   
**by** *blast*

For every encoding and every target term relation RelT there exists a source term relation RelS such that the encoding is fully abstract w.r.t. RelS and RelT.

**lemma** (in *encoding*) *fully-abstract-wrt-target-relation*:

**fixes**  $RelT :: ('procT \times 'procT) \text{ set}$   
**shows**  $\exists RelS. \text{fully-abstract } RelS RelT$

**proof** –

**define**  $RelS$  **where**  $RelS = \{(S1, S2). (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in RelT\}$

**hence** *fully-abstract*  $RelS RelT$

**by** *simp*

**thus**  $\exists RelS. \text{fully-abstract } RelS RelT$

**by** *blast*

**qed**

## 9.2 Fully Abstract Encodings

Thus, as long as we can choose one of the two relations, full abstraction is trivial. For fixed source and target term relations encodings are not trivially fully abstract. For all encodings and relations SRel and TRel we can construct a relation on the disjunctive union of source and target terms, whose reduction to source terms is SRel and whose reduction to target terms is TRel. But full abstraction ensures that each trans relation that relates source terms and their literal translations in both directions includes SRel iff it includes TRel restricted to translated source terms.

**lemma** (in *encoding*) *full-abstraction-and-trans-relation-contains-SRel-impl-TRel*:

**fixes**  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc \text{ set}$

**and**  $SRel :: ('procS \times 'procS) \text{ set}$

**and**  $TRel :: ('procT \times 'procT) \text{ set}$

**assumes** *fullAbs*: *fully-abstract*  $SRel TRel$

**and**  $encR: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

**and**  $srel: SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$

**and**  $trans: trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$

**shows**  $\forall S1\ S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel \longleftrightarrow (TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$

**proof** *auto*

**fix**  $S1\ S2$

**define**  $Rel'$  **where**  $Rel' = Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}$

**hence**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S1) \in Rel'$

**by** *simp*

**moreover** **assume**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$

**with** *fullAbs* **have**  $(S1, S2) \in SRel$

**by** *simp*

**with** *srel* *Rel'-def* **have**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel'$

**by** *simp*

**moreover** **from** *encR* *Rel'-def* **have**  $(SourceTerm\ S2, TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel'$

**by** *simp*

**ultimately** **show**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$

**using** *trans* *Rel'-def*

**unfolding** *trans-def*

**by** *blast*

**next**

**fix**  $S1\ S2$

**define**  $Rel'$  **where**  $Rel' = Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}$

**from** *encR* *Rel'-def* **have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S1 \rrbracket)) \in Rel'$

**by** *simp*

**moreover** **assume**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$

**with** *Rel'-def* **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel'$

by *simp*  
 moreover from *Rel'-def* have  $(\text{TargetTerm } (\llbracket S2 \rrbracket), \text{SourceTerm } S2) \in \text{Rel}'$   
 by *simp*  
 ultimately have  $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}$   
 using *trans Rel'-def*  
 unfolding *trans-def*  
 by *blast*  
 with *srel* have  $(S1, S2) \in \text{SRel}$   
 by *simp*  
 with *fullAbs* show  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel}$   
 by *simp*  
 qed

**lemma** (in *encoding*) *full-abstraction-and-trans-relation-contains-TRel-impl-SRel*:  
 fixes *Rel* ::  $(('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{ set}$   
 and *SRel* ::  $('procS \times 'procS) \text{ set}$   
 and *TRel* ::  $('procT \times 'procT) \text{ set}$   
 assumes *fullAbs*: *fully-abstract SRel TRel*  
 and *encR*:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$   
 and *trel*:  $\forall S1 S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel} \longleftrightarrow (\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}$   
 and *trans*:  $\text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\})$   
 shows  $\text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$

**proof** *auto*  
 fix *S1 S2*  
 define *Rel'* where  $\text{Rel}' = \text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$   
 from *encR Rel'-def* have  $(\text{SourceTerm } S1, \text{TargetTerm } (\llbracket S1 \rrbracket)) \in \text{Rel}'$   
 by *simp*  
 moreover assume  $(S1, S2) \in \text{SRel}$   
 with *fullAbs* have  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel}$   
 by *simp*  
 with *trel Rel'-def* have  $(\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}'$   
 by *simp*  
 moreover from *Rel'-def* have  $(\text{TargetTerm } (\llbracket S2 \rrbracket), \text{SourceTerm } S2) \in \text{Rel}'$   
 by *simp*  
 ultimately show  $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}$   
 using *trans Rel'-def*  
 unfolding *trans-def*  
 by *blast*

**next**  
 fix *S1 S2*  
 define *Rel'* where  $\text{Rel}' = \text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$   
 hence  $(\text{TargetTerm } (\llbracket S1 \rrbracket), \text{SourceTerm } S1) \in \text{Rel}'$   
 by *simp*  
 moreover assume  $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}$   
 with *Rel'-def* have  $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}'$   
 by *simp*  
 moreover from *encR Rel'-def* have  $(\text{SourceTerm } S2, \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}'$   
 by *simp*  
 ultimately have  $(\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}$   
 using *trans Rel'-def*  
 unfolding *trans-def*  
 by *blast*  
 with *trel* have  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel}$   
 by *simp*  
 with *fullAbs* show  $(S1, S2) \in \text{SRel}$   
 by *simp*  
 qed

**lemma** (in *encoding*) *full-abstraction-impl-trans-relation-contains-SRel-iff-TRel*:  
 fixes *Rel* ::  $(('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{ set}$   
 and *SRel* ::  $('procS \times 'procS) \text{ set}$

```

and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and encR:  ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
  and trans: trans (Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈ T P ∧ S ∈ S Q})
shows (∀ S1 S2. (⟦S1⟧, ⟦S2⟧) ∈ TRel ⟷ (TargetTerm (⟦S1⟧), TargetTerm (⟦S2⟧)) ∈ Rel)
  ⟷ (SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel})
proof
assume ∀ S1 S2. ((⟦S1⟧, ⟦S2⟧) ∈ TRel) = ((TargetTerm (⟦S1⟧), TargetTerm (⟦S2⟧)) ∈ Rel)
thus SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
  using assms full-abstraction-and-trans-relation-contains-TRel-impl-SRel[where
    SRel=SRel and TRel=TRel]
  by blast
next
assume SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
thus ∀ S1 S2. (⟦S1⟧, ⟦S2⟧) ∈ TRel ⟷ (TargetTerm (⟦S1⟧), TargetTerm (⟦S2⟧)) ∈ Rel
  using assms full-abstraction-and-trans-relation-contains-SRel-impl-TRel[where
    SRel=SRel and TRel=TRel]
  by blast
qed

```

```

lemma (in encoding) full-abstraction-impl-trans-relation-contains-SRel-iff-TRel-encRL:
fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  and SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and encR:  ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
  and encL:  ∀ S. (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel
  and trans: trans Rel
shows (∀ S1 S2. (⟦S1⟧, ⟦S2⟧) ∈ TRel ⟷ (TargetTerm (⟦S1⟧), TargetTerm (⟦S2⟧)) ∈ Rel)
  ⟷ (SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel})
proof –
from encL have Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈ T P ∧ S ∈ S Q} = Rel
  by auto
with fullAbs encR trans show ?thesis
  using full-abstraction-impl-trans-relation-contains-SRel-iff-TRel[where Rel=Rel
    and SRel=SRel and TRel=TRel]
  by simp
qed

```

Full abstraction ensures that SRel and TRel satisfy the same basic properties that can be defined on their pairs. In particular: (1) SRel is refl iff TRel reduced to translated source terms is refl (2) if the encoding is surjective then SRel is refl iff TRel is refl (3) SRel is sym iff TRel reduced to translated source terms is sym (4) SRel is trans iff TRel reduced to translated source terms is trans

**lemma** (**in** encoding) full-abstraction-impl-SRel-iff-TRel-is-refl:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
shows refl SRel ⟷ (∀ S. (⟦S⟧, ⟦S⟧) ∈ TRel)
  unfolding refl-on-def
  by (simp add: fullAbs)

```

**lemma** (**in** encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-refl:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and surj:  ∀ T. ∃ S. T = ⟦S⟧
shows refl SRel ⟷ refl TRel

```

```

proof
assume reflS: refl SRel
show refl TRel

```

```

unfolding refl-on-def
proof auto
fix T
from surj obtain S where T =  $\llbracket S \rrbracket$ 
  by blast
moreover from reflS have (S, S)  $\in$  SRel
  unfolding refl-on-def
  by simp
with fullAbs have ( $\llbracket S \rrbracket$ ,  $\llbracket S \rrbracket$ )  $\in$  TRel
  by simp
ultimately show (T, T)  $\in$  TRel
  by simp
qed
next
assume refl TRel
with fullAbs show refl SRel
  unfolding refl-on-def
  by simp
qed

```

```

lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-sym:
fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
shows sym SRel  $\longleftrightarrow$  sym  $\{(T1, T2). \exists S1 S2. T1 = \llbracket S1 \rrbracket \wedge T2 = \llbracket S2 \rrbracket \wedge (T1, T2) \in TRel\}$ 
  unfolding sym-def
  by (simp add: fullAbs, blast)

```

```

lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-sym:
fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and surj:  $\forall T. \exists S. T = \llbracket S \rrbracket$ 
shows sym SRel  $\longleftrightarrow$  sym TRel
  using fullAbs surj
  full-abstraction-impl-SRel-iff-TRel-is-sym[where SRel=SRel and TRel=TRel]
by auto

```

```

lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-trans:
fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
shows trans SRel  $\longleftrightarrow$  trans  $\{(T1, T2). \exists S1 S2. T1 = \llbracket S1 \rrbracket \wedge T2 = \llbracket S2 \rrbracket \wedge (T1, T2) \in TRel\}$ 
  unfolding trans-def
  by (simp add: fullAbs, blast)

```

```

lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-trans:
fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and surj:  $\forall T. \exists S. T = \llbracket S \rrbracket$ 
shows trans SRel  $\longleftrightarrow$  trans TRel
  using fullAbs surj
  full-abstraction-impl-SRel-iff-TRel-is-trans[where SRel=SRel and TRel=TRel]
by auto

```

Similarly, a fully abstract encoding that respects a predicate ensures the this predicate is preserved, reflected, or respected by SRel iff it is preserved, reflected, or respected by TRel.

```

lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-preserve:
fixes SRel :: ('procS  $\times$  'procS) set

```

```

and  $TRel :: ('procT \times 'procT) \text{ set}$ 
and  $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$ 
assumes  $fullAbs: \text{fully-abstract } SRel \ TRel$ 
and  $encP: \text{enc-respects-pred } Pred$ 
shows  $rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel\} \ Pred$ 
 $\longleftrightarrow rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. \llbracket SP \rrbracket \in T \ P \wedge \llbracket SQ \rrbracket \in T \ Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \ Pred$ 
proof
assume  $presS: rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel\} \ Pred$ 
show  $rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. \llbracket SP \rrbracket \in T \ P \wedge \llbracket SQ \rrbracket \in T \ Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \ Pred$ 
proof clarify
fix  $SP \ SQ$ 
assume  $Pred \ (TargetTerm \ (\llbracket SP \rrbracket))$ 
with  $encP$  have  $Pred \ (SourceTerm \ SP)$ 
by  $simp$ 
moreover assume  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
with  $fullAbs$  have  $(SP, SQ) \in SRel$ 
by  $simp$ 
ultimately have  $Pred \ (SourceTerm \ SQ)$ 
using  $presS$ 
by  $blast$ 
with  $encP$  show  $Pred \ (TargetTerm \ (\llbracket SQ \rrbracket))$ 
by  $simp$ 
qed
next
assume  $presT:$ 
 $rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. \llbracket SP \rrbracket \in T \ P \wedge \llbracket SQ \rrbracket \in T \ Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \ Pred$ 
show  $rel\text{-preserves-pred } \{(P, Q). \exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel\} \ Pred$ 
proof clarify
fix  $SP \ SQ$ 
assume  $Pred \ (SourceTerm \ SP)$ 
with  $encP$  have  $Pred \ (TargetTerm \ (\llbracket SP \rrbracket))$ 
by  $simp$ 
moreover assume  $(SP, SQ) \in SRel$ 
with  $fullAbs$  have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
by  $simp$ 
ultimately have  $Pred \ (TargetTerm \ (\llbracket SQ \rrbracket))$ 
using  $presT$ 
by  $blast$ 
with  $encP$  show  $Pred \ (SourceTerm \ SQ)$ 
by  $simp$ 
qed
qed

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-preserve:
fixes  $SRel :: ('procS \times 'procS) \text{ set}$ 
and  $TRel :: ('procT \times 'procT) \text{ set}$ 
and  $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool}$ 
assumes  $fullAbs: \text{fully-abstract } SRel \ TRel$ 
and  $encP: \text{enc-respects-binary-pred } Pred$ 
shows  $rel\text{-preserves-binary-pred } \{(P, Q). \exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel\} \ Pred$ 
 $\longleftrightarrow rel\text{-preserves-binary-pred}$ 
 $\{(P, Q). \exists SP \ SQ. \llbracket SP \rrbracket \in T \ P \wedge \llbracket SQ \rrbracket \in T \ Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \ Pred$ 
proof
assume  $presS:$ 
 $rel\text{-preserves-binary-pred } \{(P, Q). \exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel\} \ Pred$ 
show  $rel\text{-preserves-binary-pred}$ 
 $\{(P, Q). \exists SP \ SQ. \llbracket SP \rrbracket \in T \ P \wedge \llbracket SQ \rrbracket \in T \ Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \ Pred$ 
proof clarify
fix  $x \ SP \ SQ$ 
assume  $Pred \ (TargetTerm \ (\llbracket SP \rrbracket)) \ x$ 
with  $encP$  have  $Pred \ (SourceTerm \ SP) \ x$ 

```

```

    by simp
  moreover assume ( $\llbracket SP \rrbracket, \llbracket SQ \rrbracket$ )  $\in$   $TRel$ 
  with fullAbs have  $(SP, SQ) \in SRel$ 
    by simp
  ultimately have  $Pred$  ( $SourceTerm$   $SQ$ )  $x$ 
    using presS
    by blast
  with encP show  $Pred$  ( $TargetTerm$  ( $\llbracket SQ \rrbracket$ ))  $x$ 
    by simp
qed
next
assume presT:
  rel-preserves-binary-pred  $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$   $Pred$ 
show rel-preserves-binary-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$   $Pred$ 
proof clarify
  fix  $x SP SQ$ 
  assume  $Pred$  ( $SourceTerm$   $SP$ )  $x$ 
  with encP have  $Pred$  ( $TargetTerm$  ( $\llbracket SP \rrbracket$ ))  $x$ 
    by simp
  moreover assume  $(SP, SQ) \in SRel$ 
  with fullAbs have ( $\llbracket SP \rrbracket, \llbracket SQ \rrbracket$ )  $\in$   $TRel$ 
    by simp
  ultimately have  $Pred$  ( $TargetTerm$  ( $\llbracket SQ \rrbracket$ ))  $x$ 
    using presT
    by blast
  with encP show  $Pred$  ( $SourceTerm$   $SQ$ )  $x$ 
    by simp
qed
qed

lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects:
  fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
  and Pred :: ('procS, 'procT) Proc  $\Rightarrow$  bool
  assumes fullAbs: fully-abstract SRel TRel
  and encP: enc-respects-pred Pred
  shows rel-reflects-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$   $Pred$ 
   $\longleftrightarrow$  rel-reflects-pred  $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$   $Pred$ 
proof
  assume reflS: rel-reflects-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$   $Pred$ 
  show rel-reflects-pred  $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$   $Pred$ 
  proof clarify
    fix  $SP SQ$ 
    assume  $Pred$  ( $TargetTerm$  ( $\llbracket SQ \rrbracket$ ))
    with encP have  $Pred$  ( $SourceTerm$   $SQ$ )
      by simp
    moreover assume ( $\llbracket SP \rrbracket, \llbracket SQ \rrbracket$ )  $\in$   $TRel$ 
    with fullAbs have  $(SP, SQ) \in SRel$ 
      by simp
    ultimately have  $Pred$  ( $SourceTerm$   $SP$ )
      using reflS
      by blast
    with encP show  $Pred$  ( $TargetTerm$  ( $\llbracket SP \rrbracket$ ))
      by simp
  qed
next
assume reflT:
  rel-reflects-pred  $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$   $Pred$ 
show rel-reflects-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$   $Pred$ 
proof clarify
  fix  $SP SQ$ 

```



```

assume Pred (SourceTerm SQ)
with encP have Pred (TargetTerm ( $\llbracket SQ \rrbracket$ ))
  by simp
moreover assume (SP, SQ)  $\in$  SRel
with fullAbs have ( $\llbracket SP \rrbracket$ ,  $\llbracket SQ \rrbracket$ )  $\in$  TRel
  by simp
ultimately have Pred (TargetTerm ( $\llbracket SP \rrbracket$ ))
  using reflT
  by blast
with encP show Pred (SourceTerm SP)
  by simp
qed
qed

```

**lemma** (in *encoding*) *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects*:

```

fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
  and Pred :: ('procS, 'procT) Proc  $\Rightarrow$  'b  $\Rightarrow$  bool
assumes fullAbs: fully-abstract SRel TRel
  and encP: enc-respects-binary-pred Pred
shows rel-reflects-binary-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$  Pred
   $\longleftrightarrow$  rel-reflects-binary-pred
   $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$  Pred

```

**proof**

```

assume reflS:
  rel-reflects-binary-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$  Pred
show rel-reflects-binary-pred
   $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$  Pred

```

**proof** *clarify*

```

fix x SP SQ
assume Pred (TargetTerm ( $\llbracket SQ \rrbracket$ )) x
with encP have Pred (SourceTerm SQ) x
  by simp
moreover assume ( $\llbracket SP \rrbracket$ ,  $\llbracket SQ \rrbracket$ )  $\in$  TRel
with fullAbs have (SP, SQ)  $\in$  SRel
  by simp
ultimately have Pred (SourceTerm SP) x
  using reflS
  by blast
with encP show Pred (TargetTerm ( $\llbracket SP \rrbracket$ )) x
  by simp

```

**qed**

**next**

```

assume reflT:
  rel-reflects-binary-pred  $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\}$  Pred
show rel-reflects-binary-pred  $\{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\}$  Pred

```

**proof** *clarify*

```

fix x SP SQ
assume Pred (SourceTerm SQ) x
with encP have Pred (TargetTerm ( $\llbracket SQ \rrbracket$ )) x
  by simp
moreover assume (SP, SQ)  $\in$  SRel
with fullAbs have ( $\llbracket SP \rrbracket$ ,  $\llbracket SQ \rrbracket$ )  $\in$  TRel
  by simp
ultimately have Pred (TargetTerm ( $\llbracket SP \rrbracket$ )) x
  using reflT
  by blast
with encP show Pred (SourceTerm SP) x
  by simp

```

**qed**

**qed**

**lemma** (in *encoding*) *full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-respects*:  
**fixes**  $SRel :: ('procS \times 'procS) \text{ set}$   
**and**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$   
**assumes** *fullAbs*: *fully-abstract SRel TRel*  
**and** *encP*: *enc-respects-pred Pred*  
**shows**  $rel\text{-respects-pred} \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} Pred$   
 $\longleftrightarrow rel\text{-respects-pred} \{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} Pred$   
**using** *assms* *full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-preserve* [where  
 $SRel=SRel$  **and**  $TRel=TRel$  **and**  $Pred=Pred$ ]  
*full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects* [where  
 $SRel=SRel$  **and**  $TRel=TRel$  **and**  $Pred=Pred$ ]  
**by** *auto*

**lemma** (in *encoding*) *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-respects*:  
**fixes**  $SRel :: ('procS \times 'procS) \text{ set}$   
**and**  $TRel :: ('procT \times 'procT) \text{ set}$   
**and**  $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool}$   
**assumes** *fullAbs*: *fully-abstract SRel TRel*  
**and** *encP*: *enc-respects-binary-pred Pred*  
**shows**  $rel\text{-respects-binary-pred} \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} Pred$   
 $\longleftrightarrow rel\text{-respects-binary-pred} \{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} Pred$   
**using** *assms* *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-preserve* [where  
 $SRel=SRel$  **and**  $TRel=TRel$  **and**  $Pred=Pred$ ]  
*full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects* [where  
 $SRel=SRel$  **and**  $TRel=TRel$  **and**  $Pred=Pred$ ]  
**by** *auto*

### 9.3 Full Abstraction w.r.t. Preorders

If there however exists a trans relation  $Rel$  that relates source terms and their literal translations in both directions, then the encoding is fully abstract with respect to the reduction of  $Rel$  to source terms and the reduction of  $Rel$  to target terms.

**lemma** (in *encoding*) *trans-source-target-relation-impl-full-abstraction*:  
**fixes**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *enc*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$   
**and** *trans*: *trans Rel*  
**shows** *fully-abstract*  $\{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$   
 $\{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$   
**proof** *auto*  
**fix**  $S1 S2$   
**assume**  $(SourceTerm S1, SourceTerm S2) \in Rel$   
**with** *enc trans* **show**  $(TargetTerm (\llbracket S1 \rrbracket), TargetTerm (\llbracket S2 \rrbracket)) \in Rel$   
**unfolding** *trans-def*  
**by** *blast*  
**next**  
**fix**  $S1 S2$   
**assume**  $(TargetTerm (\llbracket S1 \rrbracket), TargetTerm (\llbracket S2 \rrbracket)) \in Rel$   
**with** *enc trans* **show**  $(SourceTerm S1, SourceTerm S2) \in Rel$   
**unfolding** *trans-def*  
**by** *blast*  
**qed**

**lemma** (in *encoding*) *source-target-relation-impl-full-abstraction-wrt-trans-closures*:  
**fixes**  $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$   
**assumes** *enc*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel$

**shows** *fully-abstract*  $\{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel^+\}$   
 $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel^+\}$

**proof** *auto*  
**fix**  $S1\ S2$   
**from** *enc* **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S1) \in Rel^+$   
**by** *blast*  
**moreover** **assume**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel^+$   
**ultimately** **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S2) \in Rel^+$   
**by** *simp*  
**moreover** **from** *enc* **have**  $(SourceTerm\ S2, TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel^+$   
**by** *blast*  
**ultimately** **show**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel^+$   
**by** *simp*

**next**  
**fix**  $S1\ S2$   
**from** *enc* **have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S1 \rrbracket)) \in Rel^+$   
**by** *blast*  
**moreover** **assume**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel^+$   
**ultimately** **have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel^+$   
**by** *simp*  
**moreover** **from** *enc* **have**  $(TargetTerm\ (\llbracket S2 \rrbracket), SourceTerm\ S2) \in Rel^+$   
**by** *blast*  
**ultimately** **show**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel^+$   
**by** *simp*

**qed**

**lemma** (*in encoding*) *quasi-trans-source-target-relation-impl-full-abstraction*:  
**fixes**  $Rel :: ('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc\ set$   
**and**  $SRel :: ('procS \times 'procS)\ set$   
**and**  $TRel :: ('procT \times 'procT)\ set$   
**assumes** *enc*:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$   
**and** *srel*:  $SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$   
**and** *trel*:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and** *trans*:  $\forall P\ Q\ R. (P, Q) \in Rel \wedge (Q, R) \in Rel \wedge ((P \in ProcS \wedge Q \in ProcT) \vee (P \in ProcT \wedge Q \in ProcS)) \longrightarrow (P, R) \in Rel$

**shows** *fully-abstract*  $SRel\ TRel$

**proof** *auto*  
**fix**  $S1\ S2$   
**from** *enc* **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S1) \in Rel$   
**by** *simp*  
**moreover** **assume**  $(S1, S2) \in SRel$   
**with** *srel* **have**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel$   
**by** *simp*  
**ultimately** **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S2) \in Rel$   
**using** *trans*  
**by** *blast*  
**moreover** **from** *enc* **have**  $(SourceTerm\ S2, TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$   
**by** *simp*  
**ultimately** **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$   
**using** *trans*  
**by** *blast*  
**with** *trel* **show**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$   
**by** *simp*

**next**  
**fix**  $S1\ S2$   
**from** *enc* **have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S1 \rrbracket)) \in Rel$   
**by** *simp*  
**moreover** **assume**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$   
**with** *trel* **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$   
**by** *simp*

**ultimately have** (*SourceTerm*  $S1$ , *TargetTerm* ( $\llbracket S2 \rrbracket$ ))  $\in$  *Rel*  
**using** *trans*  
**by** *blast*  
**moreover from** *enc* **have** (*TargetTerm* ( $\llbracket S2 \rrbracket$ ), *SourceTerm*  $S2$ )  $\in$  *Rel*  
**by** *simp*  
**ultimately have** (*SourceTerm*  $S1$ , *SourceTerm*  $S2$ )  $\in$  *Rel*  
**using** *trans*  
**by** *blast*  
**with** *srel* **show** ( $S1$ ,  $S2$ )  $\in$  *SRel*  
**by** *simp*  
**qed**

If an encoding is fully abstract w.r.t. *SRel* and *TRel*, then we can conclude from a pair in *indRelRTPO* or *indRelSTEQ* on a pair in *TRel* and *SRel*.

**lemma** (in *encoding*) *full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel*:

**fixes** *SRel* :: ('procS  $\times$  'procS) set  
**and** *TRel* :: ('procT  $\times$  'procT) set  
**and** *P Q* :: ('procS, 'procT) Proc  
**assumes** *fullAbs*: *fully-abstract SRel TRel*  
**and** *rel*:  $P \lesssim_{\llbracket \cdot \rrbracket} R < SRel, TRel > Q$   
**shows**  $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel^+$   
**and**  $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in TRel^*$   
**proof** –  
**have** *fullAbsT*:  $\forall S1 S2. (S1, S2) \in SRel^+ \longrightarrow (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel^+$   
**proof** *clarify*  
**fix** *S1 S2*  
**assume**  $(S1, S2) \in SRel^+$   
**thus**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel^+$   
**proof** *induct*  
**fix** *S2*  
**assume**  $(S1, S2) \in SRel$   
**with** *fullAbs* **show**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel^+$   
**by** *simp*  
**next**  
**case** (*step S2 S3*)  
**assume**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel^+$   
**moreover assume**  $(S2, S3) \in SRel$   
**with** *fullAbs* **have**  $(\llbracket S2 \rrbracket, \llbracket S3 \rrbracket) \in TRel^+$   
**by** *simp*  
**ultimately show**  $(\llbracket S1 \rrbracket, \llbracket S3 \rrbracket) \in TRel^+$   
**by** *simp*  
**qed**  
**qed**  
**with** *rel* **show**  $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel^+$   
**using** *indRelRSTPO-to-SRel-and-TRel(1)* [**where** *SRel*=*SRel* **and** *TRel*=*TRel*]  
**by** *simp*  
**show**  $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in TRel^*$   
**proof** *clarify*  
**fix** *SP TQ*  
**assume**  $SP \in S P$  **and**  $TQ \in T Q$   
**with** *rel* **obtain** *S* **where** *A1*:  $(SP, S) \in SRel^*$   
**and** *A2*:  $(\llbracket S \rrbracket, TQ) \in TRel^*$   
**using** *indRelRSTPO-to-SRel-and-TRel(2)* [**where** *SRel*=*SRel* **and** *TRel*=*TRel*]  
**by** *blast*  
**from** *A1* **have**  $SP = S \vee (SP, S) \in SRel^+$   
**using** *rtrancl-eq-or-trancl* [*of SP S SRel*]  
**by** *blast*  
**with** *fullAbsT* **have**  $(\llbracket SP \rrbracket, \llbracket S \rrbracket) \in TRel^*$   
**by** *fast*  
**from** *this A2* **show**  $(\llbracket SP \rrbracket, TQ) \in TRel^*$

by simp  
qed  
qed

**lemma** (in *encoding*) *full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel*:

fixes  $SRel :: ('procS \times 'procS) \text{ set}$   
 and  $TRel :: ('procT \times 'procT) \text{ set}$   
 and  $P \ Q :: ('procS, 'procT) \text{ Proc}$   
 assumes  $fA$ : *fully-abstract SRel TRel*  
 and  $transT$ : *trans TRel*  
 and  $reflS$ : *refl SRel*  
 and  $rel$ :  $P \sim [\cdot] \langle SRel, TRel \rangle Q$

shows  $\forall SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \longrightarrow (SP, SQ) \in SRel$   
 and  $\forall SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \longrightarrow ([SP], [SQ]) \in TRel$   
 and  $\forall SP \ TQ. SP \in S \ P \wedge TQ \in T \ Q \longrightarrow ([SP], TQ) \in TRel$   
 and  $\forall TP \ SQ. TP \in T \ P \wedge SQ \in S \ Q \longrightarrow (TP, [SQ]) \in TRel$   
 and  $\forall TP \ TQ. TP \in T \ P \wedge TQ \in T \ Q \longrightarrow (TP, TQ) \in TRel$   
 using  $rel$

**proof** *induct*

case (*encR S*)

show  $\forall SP \ SQ. SP \in S \ \text{SourceTerm } S \wedge SQ \in S \ \text{TargetTerm } ([S]) \longrightarrow (SP, SQ) \in SRel$   
 and  $\forall SP \ SQ. SP \in S \ \text{SourceTerm } S \wedge SQ \in S \ \text{TargetTerm } ([S]) \longrightarrow ([SP], [SQ]) \in TRel$   
 and  $\forall TP \ SQ. TP \in T \ \text{SourceTerm } S \wedge SQ \in S \ \text{TargetTerm } ([S]) \longrightarrow (TP, [SQ]) \in TRel$   
 and  $\forall TP \ TQ. TP \in T \ \text{SourceTerm } S \wedge TQ \in T \ \text{TargetTerm } ([S]) \longrightarrow (TP, TQ) \in TRel$   
 by *simp+*

from  $reflS \ fA$  show  $\forall SP \ TQ. SP \in S \ \text{SourceTerm } S \wedge TQ \in T \ \text{TargetTerm } ([S]) \longrightarrow ([SP], TQ) \in TRel$   
 unfolding *refl-on-def*

by *simp*

**next**

case (*encL S*)

show  $\forall SP \ SQ. SP \in S \ \text{TargetTerm } ([S]) \wedge SQ \in S \ \text{SourceTerm } S \longrightarrow (SP, SQ) \in SRel$   
 and  $\forall SP \ SQ. SP \in S \ \text{TargetTerm } ([S]) \wedge SQ \in S \ \text{SourceTerm } S \longrightarrow ([SP], [SQ]) \in TRel$   
 and  $\forall SP \ TQ. SP \in S \ \text{TargetTerm } ([S]) \wedge TQ \in T \ \text{SourceTerm } S \longrightarrow ([SP], TQ) \in TRel$   
 and  $\forall TP \ TQ. TP \in T \ \text{TargetTerm } ([S]) \wedge TQ \in T \ \text{SourceTerm } S \longrightarrow (TP, TQ) \in TRel$   
 by *simp+*

with  $reflS \ fA$  show  $\forall TP \ SQ. TP \in T \ \text{TargetTerm } ([S]) \wedge SQ \in S \ \text{SourceTerm } S \longrightarrow (TP, [SQ]) \in TRel$   
 unfolding *refl-on-def*

by *simp*

**next**

case (*source S1 S2*)

show  $\forall SP \ TQ. SP \in S \ \text{SourceTerm } S1 \wedge TQ \in T \ \text{SourceTerm } S2 \longrightarrow ([SP], TQ) \in TRel$   
 and  $\forall TP \ SQ. TP \in T \ \text{SourceTerm } S1 \wedge SQ \in S \ \text{SourceTerm } S2 \longrightarrow (TP, [SQ]) \in TRel$   
 and  $\forall TP \ TQ. TP \in T \ \text{SourceTerm } S1 \wedge TQ \in T \ \text{SourceTerm } S2 \longrightarrow (TP, TQ) \in TRel$   
 by *simp+*

assume  $(S1, S2) \in SRel$

thus  $\forall SP \ SQ. SP \in S \ \text{SourceTerm } S1 \wedge SQ \in S \ \text{SourceTerm } S2 \longrightarrow (SP, SQ) \in SRel$

by *simp*

with  $fA$  show  $\forall SP \ SQ. SP \in S \ \text{SourceTerm } S1 \wedge SQ \in S \ \text{SourceTerm } S2 \longrightarrow ([SP], [SQ]) \in TRel$

by *simp*

**next**

case (*target T1 T2*)

show  $\forall SP \ SQ. SP \in S \ \text{TargetTerm } T1 \wedge SQ \in S \ \text{TargetTerm } T2 \longrightarrow (SP, SQ) \in SRel$   
 and  $\forall SP \ SQ. SP \in S \ \text{TargetTerm } T1 \wedge SQ \in S \ \text{TargetTerm } T2 \longrightarrow ([SP], [SQ]) \in TRel$   
 and  $\forall SP \ TQ. SP \in S \ \text{TargetTerm } T1 \wedge TQ \in T \ \text{TargetTerm } T2 \longrightarrow ([SP], TQ) \in TRel$   
 and  $\forall TP \ SQ. TP \in T \ \text{TargetTerm } T1 \wedge SQ \in S \ \text{TargetTerm } T2 \longrightarrow (TP, [SQ]) \in TRel$   
 by *simp+*

assume  $(T1, T2) \in TRel$

thus  $\forall TP \ TQ. TP \in T \ \text{TargetTerm } T1 \wedge TQ \in T \ \text{TargetTerm } T2 \longrightarrow (TP, TQ) \in TRel$

by *simp*

**next**

case (*trans P Q R*)

**assume**  $A1: \forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
**and**  $A2: \forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in TRel$   
**and**  $A3: \forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow (TP, \llbracket SQ \rrbracket) \in TRel$   
**and**  $A4: \forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel$   
**and**  $A5: \forall SQ\ SR. SQ \in S\ Q \wedge SR \in S\ R \longrightarrow (\llbracket SQ \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**and**  $A6: \forall SQ\ TR. SQ \in S\ Q \wedge TR \in T\ R \longrightarrow (\llbracket SQ \rrbracket, TR) \in TRel$   
**and**  $A7: \forall TQ\ SR. TQ \in T\ Q \wedge SR \in S\ R \longrightarrow (TQ, \llbracket SR \rrbracket) \in TRel$   
**and**  $A8: \forall TQ\ TR. TQ \in T\ Q \wedge TR \in T\ R \longrightarrow (TQ, TR) \in TRel$   
**show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow (\llbracket SP \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**proof** *clarify*  
**fix**  $SP\ SR$   
**assume**  $A9: SP \in S\ P$  **and**  $A10: SR \in S\ R$   
**show**  $(\llbracket SP \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $A11: SQ \in S\ Q$   
**with**  $A1\ A9$  **have**  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
**by** *blast*  
**moreover from**  $A5\ A10\ A11$  **have**  $(\llbracket SQ \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**by** *blast*  
**ultimately show**  $(\llbracket SP \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**using** *transT*  
**unfolding** *trans-def*  
**by** *blast*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $A11: TQ \in T\ Q$   
**with**  $A2\ A9$  **have**  $(\llbracket SP \rrbracket, TQ) \in TRel$   
**by** *blast*  
**moreover from**  $A7\ A10\ A11$  **have**  $(TQ, \llbracket SR \rrbracket) \in TRel$   
**by** *blast*  
**ultimately show**  $(\llbracket SP \rrbracket, \llbracket SR \rrbracket) \in TRel$   
**using** *transT*  
**unfolding** *trans-def*  
**by** *blast*  
**qed**  
**qed**  
**with** *fA* **show**  $\forall SP\ SR. SP \in S\ P \wedge SR \in S\ R \longrightarrow (SP, SR) \in SRel$   
**by** *simp*  
**show**  $\forall SP\ TR. SP \in S\ P \wedge TR \in T\ R \longrightarrow (\llbracket SP \rrbracket, TR) \in TRel$   
**proof** *clarify*  
**fix**  $SP\ TR$   
**assume**  $A9: SP \in S\ P$  **and**  $A10: TR \in T\ R$   
**show**  $(\llbracket SP \rrbracket, TR) \in TRel$   
**proof** (*cases*  $Q$ )  
**case** (*SourceTerm*  $SQ$ )  
**assume**  $A11: SQ \in S\ Q$   
**with**  $A1\ A9$  **have**  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
**by** *blast*  
**moreover from**  $A6\ A10\ A11$  **have**  $(\llbracket SQ \rrbracket, TR) \in TRel$   
**by** *blast*  
**ultimately show**  $(\llbracket SP \rrbracket, TR) \in TRel$   
**using** *transT*  
**unfolding** *trans-def*  
**by** *blast*  
**next**  
**case** (*TargetTerm*  $TQ$ )  
**assume**  $A11: TQ \in T\ Q$   
**with**  $A2\ A9$  **have**  $(\llbracket SP \rrbracket, TQ) \in TRel$   
**by** *blast*  
**moreover from**  $A8\ A10\ A11$  **have**  $(TQ, TR) \in TRel$

```

    by blast
  ultimately show ( $\llbracket SP \rrbracket, TR$ )  $\in$   $TRel$ 
    using  $transT$ 
    unfolding  $trans-def$ 
  by blast
qed
qed
show  $\forall TP SR. TP \in TP \wedge SR \in SR \longrightarrow (TP, \llbracket SR \rrbracket) \in TRel$ 
proof clarify
  fix  $TP SR$ 
  assume  $A9: TP \in TP$  and  $A10: SR \in SR$ 
  show  $(TP, \llbracket SR \rrbracket) \in TRel$ 
  proof (cases  $Q$ )
    case (SourceTerm  $SQ$ )
    assume  $A11: SQ \in SQ$ 
    with  $A3 A9$  have  $(TP, \llbracket SQ \rrbracket) \in TRel$ 
      by blast
    moreover from  $A5 A10 A11$  have  $(\llbracket SQ \rrbracket, \llbracket SR \rrbracket) \in TRel$ 
      by blast
    ultimately show  $(TP, \llbracket SR \rrbracket) \in TRel$ 
      using  $transT$ 
      unfolding  $trans-def$ 
      by blast
  next
  case (TargetTerm  $TQ$ )
  assume  $A11: TQ \in TQ$ 
  with  $A4 A9$  have  $(TP, TQ) \in TRel$ 
    by blast
  moreover from  $A7 A10 A11$  have  $(TQ, \llbracket SR \rrbracket) \in TRel$ 
    by blast
  ultimately show  $(TP, \llbracket SR \rrbracket) \in TRel$ 
    using  $transT$ 
    unfolding  $trans-def$ 
    by blast
  qed
qed
show  $\forall TP TR. TP \in TP \wedge TR \in TR \longrightarrow (TP, TR) \in TRel$ 
proof clarify
  fix  $TP TR$ 
  assume  $A9: TP \in TP$  and  $A10: TR \in TR$ 
  show  $(TP, TR) \in TRel$ 
  proof (cases  $Q$ )
    case (SourceTerm  $SQ$ )
    assume  $A11: SQ \in SQ$ 
    with  $A3 A9$  have  $(TP, \llbracket SQ \rrbracket) \in TRel$ 
      by blast
    moreover from  $A6 A10 A11$  have  $(\llbracket SQ \rrbracket, TR) \in TRel$ 
      by blast
    ultimately show  $(TP, TR) \in TRel$ 
      using  $transT$ 
      unfolding  $trans-def$ 
      by blast
  next
  case (TargetTerm  $TQ$ )
  assume  $A11: TQ \in TQ$ 
  with  $A4 A9$  have  $(TP, TQ) \in TRel$ 
    by blast
  moreover from  $A8 A10 A11$  have  $(TQ, TR) \in TRel$ 
    by blast
  ultimately show  $(TP, TR) \in TRel$ 
    using  $transT$ 

```

```

    unfolding trans-def
  by blast
qed
qed
qed

```

If an encoding is fully abstract w.r.t. a preorder  $SRel$  on the source and a trans relation  $TRel$  on the target, then there exists a trans relation, namely  $indRelSTEQ$ , that relates source terms and their literal translations in both direction such that its reductions to source terms is  $SRel$  and its reduction to target terms is  $TRel$ .

**lemma** (in *encoding*) *full-abstraction-wrt-preorders-impl-trans-source-target-relation*:

```

fixes  $SRel :: ('procS \times 'procS) \text{ set}$ 
and  $TRel :: ('procT \times 'procT) \text{ set}$ 
assumes fullAbs: fully-abstract  $SRel$   $TRel$ 
and reflS: refl  $SRel$ 
and transT: trans  $TRel$ 
shows  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
   $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$ 
   $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
   $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
   $\wedge \text{trans}\ Rel$ 

proof –
have  $\forall S. SourceTerm\ S \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle TargetTerm\ (\llbracket S \rrbracket)$ 
   $\wedge TargetTerm\ (\llbracket S \rrbracket) \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle SourceTerm\ S$ 
using indRelSTEQ.encR[where  $SRel=SRel$  and  $TRel=TRel$ ]
  indRelSTEQ.encL[where  $SRel=SRel$  and  $TRel=TRel$ ]
by blast
moreover have  $SRel = \{(S1, S2). SourceTerm\ S1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle SourceTerm\ S2\}$ 
proof auto
fix  $S1\ S2$ 
assume  $(S1, S2) \in SRel$ 
thus  $SourceTerm\ S1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle SourceTerm\ S2$ 
by (rule indRelSTEQ.source[where  $SRel=SRel$  and  $TRel=TRel$ ])
next
fix  $S1\ S2$ 
assume  $SourceTerm\ S1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle SourceTerm\ S2$ 
with fullAbs reflS transT show  $(S1, S2) \in SRel$ 
using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(1)[where  $SRel=SRel$ 
and  $TRel=TRel$ ]
by blast
qed
moreover have  $TRel = \{(T1, T2). TargetTerm\ T1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle TargetTerm\ T2\}$ 
proof auto
fix  $T1\ T2$ 
assume  $(T1, T2) \in TRel$ 
thus  $TargetTerm\ T1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle TargetTerm\ T2$ 
by (rule indRelSTEQ.target[where  $SRel=SRel$  and  $TRel=TRel$ ])
next
fix  $T1\ T2$ 
assume  $TargetTerm\ T1 \sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle TargetTerm\ T2$ 
with fullAbs reflS transT show  $(T1, T2) \in TRel$ 
using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(5)[where  $SRel=SRel$ 
and  $TRel=TRel$ ]
by blast
qed
moreover have trans (indRelSTEQ  $SRel$   $TRel$ )
using indRelSTEQ.trans[where  $SRel=SRel$  and  $TRel=TRel$ ]
unfolding trans-def
by blast
ultimately show ?thesis

```



by *blast*  
qed

Thus an encoding is fully abstract w.r.t. a preorder  $SRel$  on the source and a trans relation  $TRel$  on the target iff there exists a trans relation that relates source terms and their literal translations in both directions and whose reduction to source/target terms is  $SRel/TRel$ .

**theorem** (in *encoding*) *fully-abstract-wrt-preorders-iff-source-target-relation-is-trans*:

**fixes**  $SRel :: ('procS \times 'procS) \text{ set}$

**and**  $TRel :: ('procT \times 'procT) \text{ set}$

**shows** (*fully-abstract*  $SRel$   $TRel \wedge refl$   $SRel \wedge trans$   $TRel$ ) =

$$\begin{aligned} & (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \\ & \quad \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \\ & \quad \wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\} \\ & \quad \wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\} \\ & \quad \wedge trans\ Rel) \end{aligned}$$

**proof** (*rule iff1*)

**assume** *fully-abstract*  $SRel$   $TRel \wedge refl$   $SRel \wedge trans$   $TRel$

**thus**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

$\wedge trans\ Rel$

**using** *full-abstraction-wrt-preorders-impl-trans-source-target-relation*[**where**  $SRel=SRel$

**and**  $TRel=TRel$ ]

by *blast*

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

$\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

$\wedge trans\ Rel$

**from this obtain**  $Rel$

**where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$

**and**  $A2: SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$

**and**  $A3: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$  **and**  $A4: trans\ Rel$

by *blast*

**hence** *fully-abstract*  $SRel$   $TRel$

**using** *trans-source-target-relation-impl-full-abstraction*[**where**  $Rel=Rel$ ]

by *blast*

**moreover have**  $refl$   $SRel$

**unfolding** *refl-on-def*

**proof** *auto*

**fix**  $S$

**from**  $A1$  **have**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

by *blast*

**moreover from**  $A1$  **have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$

by *blast*

**ultimately have**  $(SourceTerm\ S, SourceTerm\ S) \in Rel$

**using**  $A4$

**unfolding** *trans-def*

by *blast*

**with**  $A2$  **show**  $(S, S) \in SRel$

by *blast*

qed

**moreover from**  $A3$   $A4$  **have**  $trans$   $TRel$

**unfolding** *trans-def*

by *blast*

**ultimately show** *fully-abstract*  $SRel$   $TRel \wedge refl$   $SRel \wedge trans$   $TRel$

by *blast*

qed

## 9.4 Full Abstraction w.r.t. Equivalences

If there exists a relation  $Rel$  that relates source terms and their literal translations and whose sym closure is trans, then the encoding is fully abstract with respect to the reduction of the sym closure of  $Rel$  to source/target terms.

**lemma** (in *encoding*) *source-target-relation-with-trans-symcl-impl-full-abstraction:*

**fixes**  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc$  *set*

**assumes**  $enc: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

**and**  $trans: trans\ (symcl\ Rel)$

**shows** *fully-abstract*  $\{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel\}$   
 $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in symcl\ Rel\}$

**proof** *auto*

**fix**  $S1\ S2$

**from**  $enc$  **have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S1) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**moreover** **assume**  $(SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel$

**moreover** **from**  $enc$  **have**  $(SourceTerm\ S2, TargetTerm\ (\llbracket S2 \rrbracket)) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**ultimately** **show**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in symcl\ Rel$

**using**  $trans$

**unfolding**  $trans-def$

**by**  $blast$

**next**

**fix**  $S1\ S2$

**from**  $enc$  **have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S1 \rrbracket)) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**moreover** **assume**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in symcl\ Rel$

**moreover** **from**  $enc$  **have**  $(TargetTerm\ (\llbracket S2 \rrbracket), SourceTerm\ S2) \in symcl\ Rel$

**by** (*simp add: symcl-def*)

**ultimately** **show**  $(SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel$

**using**  $trans$

**unfolding**  $trans-def$

**by**  $blast$

**qed**

If an encoding is fully abstract w.r.t. the equivalences  $SRel$  and  $TRel$ , then there exists a preorder, namely  $indRelRSTPO$ , that relates source terms and their literal translations such that its reductions to source terms is  $SRel$  and its reduction to target terms is  $TRel$ .

**lemma** (in *encoding*) *fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder:*

**fixes**  $SRel :: ('procS \times 'procS)$  *set*

**and**  $TRel :: ('procT \times 'procT)$  *set*

**assumes**  $fullAbs: fully-abstract\ SRel\ TRel$

**and**  $reflT: refl\ TRel$

**and**  $symmT: sym\ TRel$

**and**  $transT: trans\ TRel$

**shows**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in symcl\ Rel\}$

$\wedge preorder\ (symcl\ Rel)$

**proof**  $-$

**from**  $fullAbs\ reflT$  **have**  $reflS: refl\ SRel$

**unfolding**  $refl-on-def$

**by**  $auto$

**from**  $fullAbs\ symmT$  **have**  $symmS: sym\ SRel$

**unfolding**  $sym-def$

**by**  $auto$

**from**  $fullAbs\ transT$  **have**  $transS: trans\ SRel$

**unfolding**  $trans-def$

**by**  $blast$

**have**  $\forall S. SourceTerm\ S \lesssim_{\llbracket \cdot \rrbracket R \langle SRel, TRel \rangle} TargetTerm\ (\llbracket S \rrbracket)$

```

    using indRelRSTPO.encR[where SRel=SRel and TRel=TRel]
  by blast
moreover
have SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl (indRelRSTPO SRel TRel)}
proof auto
  fix S1 S2
  assume (S1, S2) ∈ SRel
  thus (SourceTerm S1, SourceTerm S2) ∈ symcl (indRelRSTPO SRel TRel)
  by (simp add: symcl-def indRelRSTPO.source[where SRel=SRel and TRel=TRel])
next
  fix S1 S2
  assume (SourceTerm S1, SourceTerm S2) ∈ symcl (indRelRSTPO SRel TRel)
  moreover from transS
  have SourceTerm S1 ≲[·]R<SRel,TRel> SourceTerm S2 ⇒ (S1, S2) ∈ SRel
    using indRelRSTPO-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]
    trancl-id[of SRel]
  by blast
  moreover from symmS transS
  have SourceTerm S2 ≲[·]R<SRel,TRel> SourceTerm S1 ⇒ (S1, S2) ∈ SRel
    using indRelRSTPO-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]
    trancl-id[of SRel]
  unfolding sym-def
  by blast
  ultimately show (S1, S2) ∈ SRel
  by (auto simp add: symcl-def)
qed
moreover
have TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl (indRelRSTPO SRel TRel)}
proof auto
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus (TargetTerm T1, TargetTerm T2) ∈ symcl (indRelRSTPO SRel TRel)
  by (simp add: symcl-def indRelRSTPO.target[where SRel=SRel and TRel=TRel])
next
  fix T1 T2
  assume (TargetTerm T1, TargetTerm T2) ∈ symcl (indRelRSTPO SRel TRel)
  moreover from transT
  have TargetTerm T1 ≲[·]R<SRel,TRel> TargetTerm T2 ⇒ (T1, T2) ∈ TRel
    using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
    trancl-id[of TRel]
  by blast
  moreover from symmT transT
  have TargetTerm T2 ≲[·]R<SRel,TRel> TargetTerm T1 ⇒ (T1, T2) ∈ TRel
    using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
    trancl-id[of TRel]
  unfolding sym-def
  by blast
  ultimately show (T1, T2) ∈ TRel
  by (auto simp add: symcl-def)
qed
moreover have refl (symcl (indRelRSTPO SRel TRel))
  unfolding refl-on-def
proof auto
  fix P
  show (P, P) ∈ symcl (indRelRSTPO SRel TRel)
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈ S P
    with reflS show (P, P) ∈ symcl (indRelRSTPO SRel TRel)
      unfolding refl-on-def
      by (simp add: symcl-def indRelRSTPO.source)
  end
end

```

```

next
  case (TargetTerm TP)
  assume TP ∈ T P
  with reflT show (P, P) ∈ symcl (indRelRSTPO SRel TRel)
    unfolding refl-on-def
    by (simp add: symcl-def indRelRSTPO.target)
qed
moreover have trans (symcl (indRelRSTPO SRel TRel))
proof -
  have ∀ P Q R. P ≲[·]R<SRel,TRel> Q ∧ R ≲[·]R<SRel,TRel> Q ∧ (P, R) ∉ (indRelRSTPO SRel TRel)
    → Q ≲[·]R<SRel,TRel> P ∨ Q ≲[·]R<SRel,TRel> R
  proof clarify
    fix P Q R
    assume A1: P ≲[·]R<SRel,TRel> Q and A2: R ≲[·]R<SRel,TRel> Q
      and A3: (P, R) ∉ (indRelRSTPO SRel TRel) and A4: (Q, R) ∉ (indRelRSTPO SRel TRel)
    show Q ≲[·]R<SRel,TRel> P
    proof (cases P)
      case (SourceTerm SP)
      assume A5: SP ∈ S P
      show Q ≲[·]R<SRel,TRel> P
      proof (cases Q)
        case (SourceTerm SQ)
        assume A6: SQ ∈ S Q
        with transS A1 A5 have (SP, SQ) ∈ SRel
          using indRelRSTPO-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]
            trancl-id[of SRel]
          by blast
        with symmS A5 A6 show Q ≲[·]R<SRel,TRel> P
          unfolding sym-def
          by (simp add: indRelRSTPO.source)
      case (TargetTerm TQ)
      assume A6: TQ ∈ T Q
      show Q ≲[·]R<SRel,TRel> P
      proof (cases R)
        case (SourceTerm SR)
        assume A7: SR ∈ S R
        with fullAbs A2 A6 have ([SR], TQ) ∈ TRel*
          using full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)[where SRel=SRel
            and TRel=TRel] trancl-id[of TRel=] reflcl-of-refl-rel[of TRel]
            trancl-reflcl[of TRel]
          unfolding trans-def
          by blast
        with transT reflT have ([SR], TQ) ∈ TRel
          using trancl-id[of TRel=] reflcl-of-refl-rel[of TRel] trancl-reflcl[of TRel]
          by auto
        with symmT have (TQ, [SR]) ∈ TRel
          unfolding sym-def
          by simp
        moreover from fullAbs A1 A5 A6 have ([SP], TQ) ∈ TRel*
          using full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)[where SRel=SRel
            and TRel=TRel]
          unfolding trans-def
          by blast
        with transT reflT have ([SP], TQ) ∈ TRel
          using trancl-id[of TRel=] reflcl-of-refl-rel[of TRel] trancl-reflcl[of TRel]
          by auto
        ultimately have ([SP], [SR]) ∈ TRel
          using transT
          unfolding trans-def

```

```

    by blast
  with fullAbs have (SP, SR) ∈ SRel
    by simp
  with A3 A5 A7 show ?thesis
    by (simp add: indRelRSTPO.source)
next
  case (TargetTerm TR)
  assume A7: TR ∈ T R
  with transT A2 A6 have (TR, TQ) ∈ TRel
    using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
      trancl-id[of TRel]
    by blast
  with symmT have (TQ, TR) ∈ TRel
    unfolding sym-def
    by simp
  with A4 A6 A7 show ?thesis
    by (simp add: indRelRSTPO.target)
qed
qed
next
  case (TargetTerm TP)
  assume A5: TP ∈ T P
  show Q ≲[·]R<SRel, TRel> P
  proof (cases Q)
    case (SourceTerm SQ)
    assume SQ ∈ S Q
    with A1 A5 show ?thesis
      using indRelRSTPO-to-SRel-and-TRel(3)[where SRel=SRel and TRel=TRel]
      by blast
  next
    case (TargetTerm TQ)
    assume A6: TQ ∈ T Q
    with transT A1 A5 have (TP, TQ) ∈ TRel
      using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
      trancl-id[of TRel]
      by blast
    with symmT have (TQ, TP) ∈ TRel
      unfolding sym-def
      by simp
    with A5 A6 show Q ≲[·]R<SRel, TRel> P
      by (simp add: indRelRSTPO.target)
  qed
qed
qed
moreover
have ∀ P Q R. P ≲[·]R<SRel, TRel> Q ∧ P ≲[·]R<SRel, TRel> R ∧ (Q, R) ∉ (indRelRSTPO SRel TRel)
  → Q ≲[·]R<SRel, TRel> P ∨ R ≲[·]R<SRel, TRel> P
proof clarify
  fix P Q R
  assume A1: P ≲[·]R<SRel, TRel> Q and A2: P ≲[·]R<SRel, TRel> R
    and A3: (Q, R) ∉ (indRelRSTPO SRel TRel) and A4: (R, P) ∉ (indRelRSTPO SRel TRel)
  show Q ≲[·]R<SRel, TRel> P
  proof (cases P)
    case (SourceTerm SP)
    assume A5: SP ∈ S P
    show Q ≲[·]R<SRel, TRel> P
    proof (cases Q)
      case (SourceTerm SQ)
      assume A6: SQ ∈ S Q
      with transS A1 A5 have (SP, SQ) ∈ SRel
        using indRelRSTPO-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]

```

```

      trancl-id[of SRel]
    by blast
  with symmS A5 A6 show  $Q \lesssim[\cdot]R \langle SRel, TRel \rangle P$ 
    unfolding sym-def
    by (simp add: indRelRSTPO.source)
next
case (TargetTerm TQ)
assume A6:  $TQ \in T Q$ 
show  $Q \lesssim[\cdot]R \langle SRel, TRel \rangle P$ 
proof (cases R)
  case (SourceTerm SR)
  assume A7:  $SR \in S R$ 
  with transS A2 A5 have  $(SP, SR) \in SRel$ 
    using indRelRSTPO-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]
    trancl-id[of SRel]
  by blast
  with symmS have  $(SR, SP) \in SRel$ 
    unfolding sym-def
    by simp
  with A4 A5 A7 show ?thesis
    by (simp add: indRelRSTPO.source)
next
case (TargetTerm TR)
from fullAbs A1 A5 A6 have  $(\llbracket SP \rrbracket, TQ) \in TRel^*$ 
  using full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)[where SRel=SRel and
    TRel=TRel]
  unfolding trans-def
  by blast
with transT reflT have  $(\llbracket SP \rrbracket, TQ) \in TRel$ 
  using trancl-id[of TRel=] reflcl-of-refl-rel[of TRel] trancl-reflcl[of TRel]
  by auto
with symmT have  $(TQ, \llbracket SP \rrbracket) \in TRel$ 
  unfolding sym-def
  by simp
moreover assume A7:  $TR \in T R$ 
with fullAbs A2 A5 have  $(\llbracket SP \rrbracket, TR) \in TRel^*$ 
  using full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)[where SRel=SRel and
    TRel=TRel]
  unfolding trans-def
  by blast
with transT reflT have  $(\llbracket SP \rrbracket, TR) \in TRel$ 
  using trancl-id[of TRel=] reflcl-of-refl-rel[of TRel] trancl-reflcl[of TRel]
  by auto
ultimately have  $(TQ, TR) \in TRel$ 
  using transT
  unfolding trans-def
  by blast
with A3 A6 A7 show ?thesis
  by (simp add: indRelRSTPO.target)
qed
qed
next
case (TargetTerm TP)
assume A5:  $TP \in T P$ 
show  $Q \lesssim[\cdot]R \langle SRel, TRel \rangle P$ 
proof (cases Q)
  case (SourceTerm SQ)
  assume SQ  $\in S Q$ 
  with A1 A5 show ?thesis
    using indRelRSTPO-to-SRel-and-TRel(3)[where SRel=SRel and TRel=TRel]
    by blast

```

```

next
  case (TargetTerm TQ)
  assume A6: TQ ∈ T Q
  with transT A1 A5 have (TP, TQ) ∈ TRel
    using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
      trancl-id[of TRel]
    by blast
  with symmT have (TQ, TP) ∈ TRel
    unfolding sym-def
    by simp
  with A5 A6 show Q ≲[·]R<SRel,TRel> P
    by (simp add: indRelRSTPO.target)
qed
qed
qed
moreover from reflS reflT have refl (indRelRSTPO SRel TRel)
  using indRelRSTPO-refl[where SRel=SRel and TRel=TRel]
  by blast
moreover have trans (indRelRSTPO SRel TRel)
  using indRelRSTPO.trans[where SRel=SRel and TRel=TRel]
  unfolding trans-def
  by blast
ultimately show trans (symcl (indRelRSTPO SRel TRel))
  using symm-closure-of-preorder-is-trans[where Rel=indRelRSTPO SRel TRel]
  by blast
qed
ultimately show ?thesis
  unfolding preorder-on-def
  by blast
qed

lemma (in encoding) fully-abstract-impl-symcl-source-target-relation-is-preorder:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes fullAbs: fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+)
  shows ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
    ∧ ((symcl (SRel=))+) = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ ((symcl (TRel=))+) = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
proof -
  have refl ((symcl (TRel=))+)
    using refl-symm-trans-closure-is-symm-refl-trans-closure[of TRel]
      refl-rtrancl[of TRel]
    unfolding sym-def refl-on-def
    by auto
  moreover have sym ((symcl (TRel=))+)
    using sym-symcl[of TRel=] sym-trancl[of symcl (TRel=)]
    by simp
  moreover have trans ((symcl (TRel=))+)
    by simp
  ultimately show ?thesis
    using fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder[where
      SRel=(symcl (SRel=))+ and TRel=(symcl (TRel=))+] fullAbs
      refl-symm-closure-is-symm-refl-closure
    unfolding preorder-on-def
    by blast
qed

lemma (in encoding) fully-abstract-wrt-preorders-impl-source-target-relation-is-trans:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set

```

```

assumes fullAbs: fully-abstract SRel TRel
shows  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
   $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
   $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
   $\wedge ((refl\ SRel \wedge trans\ TRel)$ 
     $\longleftrightarrow trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}))$ 
proof –
  define Rel where Rel = (indRelSTEQ SRel TRel) – ( $\{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}$ 
     $\cup \{(P, Q). \exists S1\ S2. S1 \in S\ P \wedge S2 \in S\ Q \wedge (S1, S2) \notin SRel\}$ 
     $\cup \{(P, Q). \exists T1\ T2. T1 \in T\ P \wedge T2 \in T\ Q \wedge (T1, T2) \notin TRel\}$ )
  from Rel-def have  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
    by (simp add: indRelSTEQ.encR[where SRel=SRel and TRel=TRel])
  moreover from Rel-def have SRel =  $\{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
proof auto
  fix S1 S2
  assume (S1, S2)  $\in SRel$ 
  thus SourceTerm S1  $\sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle$  SourceTerm S2
    by (simp add: indRelSTEQ.source[where SRel=SRel and TRel=TRel])
qed
  moreover from Rel-def have TRel =  $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
proof auto
  fix T1 T2
  assume (T1, T2)  $\in TRel$ 
  thus TargetTerm T1  $\sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle$  TargetTerm T2
    by (simp add: indRelSTEQ.target[where SRel=SRel and TRel=TRel])
qed
  moreover
  have (refl SRel  $\wedge$  trans TRel)  $\longleftrightarrow trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
proof (rule iffI, erule conjE)
  assume reflS: refl SRel and transT: trans TRel
  have Rel  $\cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} = indRelSTEQ\ SRel\ TRel$ 
proof (auto simp add: Rel-def)
  fix S
  show TargetTerm ( $\llbracket S \rrbracket$ )  $\sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle$  SourceTerm S
    by (rule indRelSTEQ.encL)
  next
  fix S1 S2
  assume SourceTerm S1  $\sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle$  SourceTerm S2
  with fullAbs reflS transT have (S1, S2)  $\in SRel$ 
    using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(1)[where
      SRel=SRel and TRel=TRel]
    by blast
  moreover assume (S1, S2)  $\notin SRel$ 
  ultimately show False
    by simp
  next
  fix T1 T2
  assume TargetTerm T1  $\sim \llbracket \cdot \rrbracket \langle SRel, TRel \rangle$  TargetTerm T2
  with fullAbs reflS transT have (T1, T2)  $\in TRel$ 
    using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(5)[where
      SRel=SRel and TRel=TRel]
    by blast
  moreover assume (T1, T2)  $\notin TRel$ 
  ultimately show False
    by simp
qed
thus trans (Rel  $\cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
  using indRelSTEQ-trans[where SRel=SRel and TRel=TRel]
  unfolding trans-def
  by blast
next

```



```

assume transR: trans (Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ )
show refl SRel  $\wedge$  trans TRel
  unfolding trans-def refl-on-def
proof auto
  fix S
  from Rel-def have (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ 
    by (simp add: indRelSTEQ.encR)
  moreover have (TargetTerm ( $\llbracket S \rrbracket$ ), SourceTerm S)  $\in$  Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ 
    by simp
  ultimately have (SourceTerm S, SourceTerm S)  $\in$  Rel
    using transR
    unfolding trans-def
    by blast
  with Rel-def show (S, S)  $\in$  SRel
    by simp
next
  fix TP TQ TR
  assume (TP, TQ)  $\in$  TRel
  with Rel-def have (TargetTerm TP, TargetTerm TQ)  $\in$  Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ 
    by (simp add: indRelSTEQ.target)
  moreover assume (TQ, TR)  $\in$  TRel
  with Rel-def have (TargetTerm TQ, TargetTerm TR)  $\in$  Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ 
    by (simp add: indRelSTEQ.target)
  ultimately have (TargetTerm TP, TargetTerm TR)  $\in$  Rel
    using transR
    unfolding trans-def
    by blast
  with Rel-def show (TP, TR)  $\in$  TRel
    by simp
qed
qed
ultimately show ?thesis
  by blast
qed

```

**lemma** (*in encoding*) *fully-abstract-wrt-preorders-impl-source-target-relation-is-trans-B*:

```

fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and reflT: refl TRel
  and transT: trans TRel
shows  $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)
   $\wedge$  SRel =  $\{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$ 
   $\wedge$  TRel =  $\{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$ 
   $\wedge$  trans (Rel  $\cup$   $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ )
proof –
  define Rel where Rel = (indRelSTEQ SRel TRel) –  $\{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}$ 
  from fullAbs reflT have reflS: refl SRel
    unfolding refl-on-def
    by auto
  from Rel-def have  $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel
    by (simp add: indRelSTEQ.encR [where SRel=SRel and TRel=TRel])
  moreover from Rel-def have SRel =  $\{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$ 
proof auto
  fix S1 S2
  assume (S1, S2)  $\in$  SRel
  thus SourceTerm S1  $\sim$   $\llbracket \cdot \rrbracket$   $\langle$  SRel, TRel  $\rangle$  SourceTerm S2
    by (simp add: indRelSTEQ.source [where SRel=SRel and TRel=TRel])
next
  fix S1 S2
  assume SourceTerm S1  $\sim$   $\llbracket \cdot \rrbracket$   $\langle$  SRel, TRel  $\rangle$  SourceTerm S2

```

```

with fullAbs transT reflS show (S1, S2) ∈ SRel
  using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(1)[where SRel=SRel
    and TRel=TRel]
  by blast
qed
moreover from Rel-def have TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
proof auto
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1 ~[·]<SRel,TRel> TargetTerm T2
    by (simp add: indRelSTEQ.target[where SRel=SRel and TRel=TRel])
next
  fix T1 T2
  assume TargetTerm T1 ~[·]<SRel,TRel> TargetTerm T2
  with fullAbs transT reflS show (T1, T2) ∈ TRel
    using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(5)[where SRel=SRel
      and TRel=TRel]
    by blast
qed
moreover from Rel-def have Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q} = indRelSTEQ SRel TRel
  by (auto simp add: indRelSTEQ.encL)
hence trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
  using indRelSTEQ.trans[where SRel=SRel and TRel=TRel]
  unfolding trans-def
  by auto
ultimately show ?thesis
  by blast
qed

```

Thus an encoding is fully abstract w.r.t. an equivalence SRel on the source and an equivalence TRel on the target iff there exists a relation that relates source terms and their literal translations, whose sym closure is a preorder such that the reduction of this sym closure to source/target terms is SRel/TRel.

**lemma** (in encoding) fully-abstract-wrt-equivalences-iff-symcl-source-target-relation-is-preorder:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
shows (fully-abstract SRel TRel ∧ equivalence TRel) =
  (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel))
proof (rule iffI)
  assume fully-abstract SRel TRel ∧ equivalence TRel
  thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
    using fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder[where
      SRel=SRel and TRel=TRel]
    unfolding equiv-def
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
  from this obtain Rel
  where   ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
    and   SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    and A1: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    and A2: preorder (symcl Rel)

```

```

  by blast
hence A5: fully-abstract SRel TRel
  using source-target-relation-with-trans-symcl-impl-full-abstraction[where Rel=Rel]
  unfolding preorder-on-def
  by blast
moreover have equivalence TRel
  unfolding trans-def equiv-def sym-def refl-on-def
proof auto
  fix T
  from A1 A2 show (T, T) ∈ TRel
    unfolding preorder-on-def refl-on-def
    by blast
next
  fix T1 T2
  assume (T1, T2) ∈ TRel
  with A1 show (T2, T1) ∈ TRel
    by (auto simp add: symcl-def)
next
  fix T1 T2 T3
  assume (T1, T2) ∈ TRel and (T2, T3) ∈ TRel
  with A1 A2 show (T1, T3) ∈ TRel
    unfolding trans-def preorder-on-def
    by blast
qed
ultimately show fully-abstract SRel TRel ∧ equivalence TRel
  by blast
qed

lemma (in encoding) fully-abstract-iff-symcl-source-target-relation-is-preorder:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  shows fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+) =
    (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
      ∧ (symcl (SRel=))+ = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
      ∧ (symcl (TRel=))+ = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
      ∧ preorder (symcl Rel))
proof (rule iffI)
  assume fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+)
  thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
    ∧ (symcl (SRel=))+ = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ (symcl (TRel=))+ = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
    using fully-abstract-impl-symcl-source-target-relation-is-preorder[where SRel=SRel and
      TRel=TRel]
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
    ∧ (symcl (SRel=))+ = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ (symcl (TRel=))+ = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
  from this obtain Rel
  where   ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
  and     (symcl (SRel=))+ = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
  and A1: (symcl (TRel=))+ = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
  and A2: preorder (symcl Rel)
  by blast
  thus fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+)
    using source-target-relation-with-trans-symcl-impl-full-abstraction[where Rel=Rel]
    unfolding preorder-on-def
    by blast
qed

```

## 9.5 Full Abstraction without Relating Translations to their Source Terms

Let  $Rel$  be the result of removing from  $indRelSTEQ$  all pairs of two source or two target terms that are not contained in  $SRel$  or  $TRel$ . Then a fully abstract encoding ensures that  $Rel$  is trans iff  $SRel$  is refl and  $TRel$  is trans.

**lemma** (in *encoding*) *full-abstraction-impl-indRelSTEQ-is-trans*:

```

fixes  $SRel :: ('procS \times 'procS) \text{ set}$ 
and  $TRel :: ('procT \times 'procT) \text{ set}$ 
and  $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$ 
assumes fullAbs: fully-abstract SRel TRel
and rel:  $Rel = ((indRelSTEQ\ SRel\ TRel)$ 
   $- \{(P, Q). (P \in ProcS \wedge Q \in ProcS) \vee (P \in ProcT \wedge Q \in ProcT)\}$ 
   $\cup \{(P, Q). (\exists SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \wedge (SP, SQ) \in SRel)$ 
     $\vee (\exists TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \wedge (TP, TQ) \in TRel)\}$ 

```

**shows**  $(refl\ SRel \wedge trans\ TRel) = trans\ Rel$

**unfolding** *trans-def*

**proof** *auto*

**fix**  $P\ Q\ R$

**assume**  $A1$ : *refl SRel* **and**  $A2$ :  $\forall x\ y. (x, y) \in TRel \longrightarrow (\forall z. (y, z) \in TRel \longrightarrow (x, z) \in TRel)$

**and**  $A3$ :  $(P, Q) \in Rel$  **and**  $A4$ :  $(Q, R) \in Rel$

**from** *fullAbs rel* **have**  $A5$ :  $\forall SP\ SQ. (SourceTerm\ SP, SourceTerm\ SQ) \in Rel \longrightarrow ([SP], [SQ]) \in TRel$

**by** *simp*

**from** *rel* **have**  $A6$ :  $\forall TP\ TQ. (TargetTerm\ TP, TargetTerm\ TQ) \in Rel \longrightarrow (TP, TQ) \in TRel$

**by** *simp*

**have**  $A7$ :  $\forall SP\ TQ. (SourceTerm\ SP, TargetTerm\ TQ) \in Rel \longrightarrow ([SP], TQ) \in TRel$

**proof** *clarify*

**fix**  $SP\ TQ$

**assume**  $(SourceTerm\ SP, TargetTerm\ TQ) \in Rel$

**with** *rel* **have**  $SourceTerm\ SP \sim[\cdot] \langle SRel, TRel \rangle TargetTerm\ TQ$

**by** *simp*

**with**  $A1\ A2$  *fullAbs* **show**  $([SP], TQ) \in TRel$

**using** *full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(3)* [**where**  
 $SRel = SRel$  **and**  $TRel = TRel$ ]

**unfolding** *trans-def*

**by** *blast*

**qed**

**have**  $A8$ :  $\forall TP\ SQ. (TargetTerm\ TP, SourceTerm\ SQ) \in Rel \longrightarrow (TP, [SQ]) \in TRel$

**proof** *clarify*

**fix**  $TP\ SQ$

**assume**  $(TargetTerm\ TP, SourceTerm\ SQ) \in Rel$

**with** *rel* **have**  $TargetTerm\ TP \sim[\cdot] \langle SRel, TRel \rangle SourceTerm\ SQ$

**by** *simp*

**with**  $A1\ A2$  *fullAbs* **show**  $(TP, [SQ]) \in TRel$

**using** *full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(4)* [**where**  
 $SRel = SRel$  **and**  $TRel = TRel$ ]

**unfolding** *trans-def*

**by** *blast*

**qed**

**show**  $(P, R) \in Rel$

**proof** (*cases P*)

**case**  $(SourceTerm\ SP)$

**assume**  $A9$ :  $SP \in S\ P$

**show**  $(P, R) \in Rel$

**proof** (*cases Q*)

**case**  $(SourceTerm\ SQ)$

**assume**  $A10$ :  $SQ \in S\ Q$

**with**  $A3\ A5\ A9$  **have**  $A11$ :  $([SP], [SQ]) \in TRel$

**by** *simp*

**show**  $(P, R) \in Rel$

**proof** (*cases R*)

```

case (SourceTerm SR)
assume A12: SR ∈ S R
with A4 A5 A10 have ([[SQ]], [[SR]]) ∈ TRel
  by simp
with A2 A11 have ([[SP]], [[SR]]) ∈ TRel
  by blast
with fullAbs have (SP, SR) ∈ SRel
  by simp
with rel A9 A12 show (P, R) ∈ Rel
  by simp
next
case (TargetTerm TR)
assume A12: TR ∈ T R
from A9 have P ∼[·]<SRel,TRel> TargetTerm ([[SP]])
  by (simp add: indRelSTEQ.encR)
moreover from A4 A7 A10 A12 have ([[SQ]], TR) ∈ TRel
  by simp
with A2 A11 have ([[SP]], TR) ∈ TRel
  by blast
with A12 have TargetTerm ([[SP]]) ∼[·]<SRel,TRel> R
  by (simp add: indRelSTEQ.target)
ultimately have P ∼[·]<SRel, TRel> R
  by (rule indRelSTEQ.trans)
with rel A9 A12 show (P, R) ∈ Rel
  by simp
qed
next
case (TargetTerm TQ)
assume A10: TQ ∈ T Q
with A3 A7 A9 have A11: ([[SP]], TQ) ∈ TRel
  by simp
show (P, R) ∈ Rel
proof (cases R)
case (SourceTerm SR)
assume A12: SR ∈ S R
with A4 A8 A10 have (TQ, [[SR]]) ∈ TRel
  by simp
with A2 A11 have ([[SP]], [[SR]]) ∈ TRel
  by blast
with fullAbs have (SP, SR) ∈ SRel
  by simp
with rel A9 A12 show (P, R) ∈ Rel
  by simp
next
case (TargetTerm TR)
assume A12: TR ∈ T R
from A9 have P ∼[·]<SRel,TRel> TargetTerm ([[SP]])
  by (simp add: indRelSTEQ.encR)
moreover from A4 A6 A10 A12 have (TQ, TR) ∈ TRel
  by simp
with A2 A11 have ([[SP]], TR) ∈ TRel
  by blast
with A12 have TargetTerm ([[SP]]) ∼[·]<SRel,TRel> R
  by (simp add: indRelSTEQ.target)
ultimately have P ∼[·]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
with A9 A12 rel show (P, R) ∈ Rel
  by simp
qed
qed
next

```

**case** (*TargetTerm TP*)  
**assume** *A9: TP ∈ T P*  
**show**  $(P, R) \in Rel$   
**proof** (*cases Q*)  
  **case** (*SourceTerm SQ*)  
  **assume** *A10: SQ ∈ S Q*  
  **with** *A3 A8 A9* **have** *A11: (TP, [SQ]) ∈ TRel*  
  **by** *simp*  
  **show**  $(P, R) \in Rel$   
  **proof** (*cases R*)  
    **case** (*SourceTerm SR*)  
    **assume** *A12: SR ∈ S R*  
    **with** *A4 A5 A10* **have**  $([SQ], [SR]) \in TRel$   
    **by** *simp*  
    **with** *A2 A11* **have**  $(TP, [SR]) \in TRel$   
    **by** *blast*  
    **with** *A9* **have**  $P \sim [\cdot] \langle SRel, TRel \rangle TargetTerm ([SR])$   
    **by** (*simp add: indRelSTEQ.target*)  
    **moreover from** *A12* **have**  $TargetTerm ([SR]) \sim [\cdot] \langle SRel, TRel \rangle R$   
    **by** (*simp add: indRelSTEQ.encL*)  
    **ultimately have**  $P \sim [\cdot] \langle SRel, TRel \rangle R$   
    **by** (*rule indRelSTEQ.trans*)  
    **with rel** *A9 A12* **show**  $(P, R) \in Rel$   
    **by** *simp*  
  **next**  
  **case** (*TargetTerm TR*)  
  **assume** *A12: TR ∈ T R*  
  **with** *A4 A7 A10* **have**  $([SQ], TR) \in TRel$   
  **by** *simp*  
  **with** *A2 A11* **have**  $(TP, TR) \in TRel$   
  **by** *blast*  
  **with rel** *A9 A12* **show**  $(P, R) \in Rel$   
  **by** *simp*  
**qed**  
**next**  
  **case** (*TargetTerm TQ*)  
  **assume** *A10: TQ ∈ T Q*  
  **with** *A3 A6 A9* **have** *A11: (TP, TQ) ∈ TRel*  
  **by** *simp*  
  **show**  $(P, R) \in Rel$   
  **proof** (*cases R*)  
    **case** (*SourceTerm SR*)  
    **assume** *A12: SR ∈ S R*  
    **with** *A4 A8 A10* **have**  $(TQ, [SR]) \in TRel$   
    **by** *simp*  
    **with** *A2 A11* **have**  $(TP, [SR]) \in TRel$   
    **by** *blast*  
    **with** *A9* **have**  $P \sim [\cdot] \langle SRel, TRel \rangle TargetTerm ([SR])$   
    **by** (*simp add: indRelSTEQ.target*)  
    **moreover from** *A12* **have**  $TargetTerm ([SR]) \sim [\cdot] \langle SRel, TRel \rangle R$   
    **by** (*simp add: indRelSTEQ.encL*)  
    **ultimately have**  $P \sim [\cdot] \langle SRel, TRel \rangle R$   
    **by** (*rule indRelSTEQ.trans*)  
    **with rel** *A9 A12* **show**  $(P, R) \in Rel$   
    **by** *simp*  
  **next**  
  **case** (*TargetTerm TR*)  
  **assume** *A12: TR ∈ T R*  
  **with** *A4 A6 A10* **have**  $(TQ, TR) \in TRel$   
  **by** *simp*  
  **with** *A2 A11* **have**  $(TP, TR) \in TRel$

```

    by blast
  with A9 A12 rel show  $(P, R) \in Rel$ 
  by simp
qed
qed
qed
next
assume  $B: \forall x y. (x, y) \in Rel \longrightarrow (\forall z. (y, z) \in Rel \longrightarrow (x, z) \in Rel)$ 
thus refl SRel
  unfolding refl-on-def
proof auto
  fix S
  from rel have  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
  by (simp add: indRelSTEQ.encR)
  moreover from rel have  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$ 
  by (simp add: indRelSTEQ.encL)
  ultimately have  $(SourceTerm\ S, SourceTerm\ S) \in Rel$ 
  using B
  by blast
  with rel show  $(S, S) \in SRel$ 
  by simp
qed
next
fix TP TQ TR
assume  $\forall x y. (x, y) \in Rel \longrightarrow (\forall z. (y, z) \in Rel \longrightarrow (x, z) \in Rel)$ 
moreover assume  $(TP, TQ) \in TRel$ 
with rel have  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$ 
  by simp
moreover assume  $(TQ, TR) \in TRel$ 
with rel have  $(TargetTerm\ TQ, TargetTerm\ TR) \in Rel$ 
  by simp
ultimately have  $(TargetTerm\ TP, TargetTerm\ TR) \in Rel$ 
  by blast
with rel show  $(TP, TR) \in TRel$ 
  by simp
qed

```

Whenever an encoding induces a trans relation that includes SRel and TRel and relates source terms to their literal translations in both directions, the encoding is fully abstract w.r.t. SRel and TRel.

```

lemma (in encoding) trans-source-target-relation-impl-fully-abstract:
  fixes Rel ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc)$  set
  and SRel ::  $(\text{'procS} \times \text{'procS})$  set
  and TRel ::  $(\text{'procT} \times \text{'procT})$  set
  assumes enc:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
     $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$ 
  and srel:  $SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
  and trel:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
  and trans: trans Rel
  shows fully-abstract SRel TRel
proof auto
  fix S1 S2
  assume  $(S1, S2) \in SRel$ 
  with srel have  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel$ 
  by simp
  with enc trans have  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel$ 
  unfolding trans-def
  by blast
  with trel show  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$ 
  by simp
next

```

```

fix  $S1\ S2$ 
assume  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$ 
with  $trel$  have  $(TargetTerm(\llbracket S1 \rrbracket), TargetTerm(\llbracket S2 \rrbracket)) \in Rel$ 
  by  $simp$ 
with  $enc\ trans$  have  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel$ 
  unfolding  $trans-def$ 
  by  $blast$ 
with  $srel$  show  $(S1, S2) \in SRel$ 
  by  $simp$ 
qed

```

Assume TRel is a preorder. Then an encoding is fully abstract w.r.t. SRel and TRel iff there exists a relation that relates at least all source terms to their literal translations, includes SRel and TRel, and whose union with the relation that relates exactly all literal translations to their source terms is trans.

**lemma** (in *encoding*) *source-target-relation-with-trans-impl-full-abstraction*:

```

fixes  $Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc$  set
assumes  $enc: \forall S. (SourceTerm\ S, TargetTerm(\llbracket S \rrbracket)) \in Rel$ 
  and  $trans: trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
shows  $fully-abstract\ \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
   $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 

```

**proof** *auto*

```

fix  $S1\ S2$ 
define  $Rel'$  where  $Rel' = Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}$ 
from  $Rel'-def$  have  $(TargetTerm(\llbracket S1 \rrbracket), SourceTerm\ S1) \in Rel'$ 
  by  $simp$ 
moreover assume  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel$ 
with  $Rel'-def$  have  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel'$ 
  by  $simp$ 
moreover from  $enc\ Rel'-def$  have  $(SourceTerm\ S2, TargetTerm(\llbracket S2 \rrbracket)) \in Rel'$ 
  by  $simp$ 
ultimately show  $(TargetTerm(\llbracket S1 \rrbracket), TargetTerm(\llbracket S2 \rrbracket)) \in Rel$ 
  using  $trans\ Rel'-def$ 
  unfolding  $trans-def$ 
  by  $blast$ 

```

**next**

```

fix  $S1\ S2$ 
define  $Rel'$  where  $Rel' = Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}$ 
from  $enc\ Rel'-def$  have  $(SourceTerm\ S1, TargetTerm(\llbracket S1 \rrbracket)) \in Rel'$ 
  by  $simp$ 
moreover assume  $(TargetTerm(\llbracket S1 \rrbracket), TargetTerm(\llbracket S2 \rrbracket)) \in Rel$ 
with  $Rel'-def$  have  $(TargetTerm(\llbracket S1 \rrbracket), TargetTerm(\llbracket S2 \rrbracket)) \in Rel'$ 
  by  $simp$ 
moreover from  $Rel'-def$  have  $(TargetTerm(\llbracket S2 \rrbracket), SourceTerm\ S2) \in Rel'$ 
  by  $simp$ 
ultimately show  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel$ 
  using  $trans\ Rel'-def$ 
  unfolding  $trans-def$ 
  by  $blast$ 

```

**qed**

**lemma** (in *encoding*) *fully-abstract-wrt-preorders-iff-source-target-relation-is-transB*:

```

fixes  $SRel :: ('procS \times 'procS)$  set
  and  $TRel :: ('procT \times 'procT)$  set
assumes  $preord: preorder\ TRel$ 
shows  $fully-abstract\ SRel\ TRel =$ 
   $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm(\llbracket S \rrbracket)) \in Rel)$ 
     $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
     $\wedge trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}))$ 

```



```

proof (rule iffI)
  assume fully-abstract SRel TRel
  with preord show  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
     $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
     $\wedge trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
    using fully-abstract-wrt-preorders-impl-source-target-relation-is-trans[where SRel=SRel
      and TRel=TRel]
    unfolding preorder-on-def refl-on-def
  by auto
next
  assume  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
     $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
     $\wedge trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
  from this obtain Rel
    where  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
    and  $SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
    and  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
    and  $trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
  by blast
  thus fully-abstract SRel TRel
    using source-target-relation-with-trans-impl-full-abstraction[where Rel=Rel]
  by blast
qed

```

The same holds if to obtain transitivity the union may contain additional pairs that do neither relate two source nor two target terms.

**lemma** (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-union-is-trans:

```

fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
shows (fully-abstract SRel TRel  $\wedge$  refl SRel  $\wedge$  trans TRel) =
  ( $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
     $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
     $\wedge (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \longleftrightarrow Q \in ProcT)$ 
     $\wedge trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel')$ )
proof (rule iffI, (erule conjE)+)
  assume fully-abstract SRel TRel and refl SRel and trans TRel
  from this obtain Rel where A1:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ 
    and A2:  $SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
    and A3:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
    and A4:  $trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$ 
    using fully-abstract-wrt-preorders-impl-source-target-relation-is-trans[where SRel=SRel
      and TRel=TRel]
    by blast
  have  $\forall (P, Q) \in \{\}. P \in ProcS \longleftrightarrow Q \in ProcT$ 
    by simp
  moreover from A4 have  $trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup \{\})$ 
    unfolding trans-def
    by blast
  ultimately show  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
     $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$ 
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$ 
     $\wedge (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \longleftrightarrow Q \in ProcT)$ 
     $\wedge trans\ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel')$ 
    using A1 A2 A3
    by blast
next
  assume  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 

```

$\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \longleftrightarrow Q \in ProcT)$   
 $\wedge trans (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel')$

**from this obtain**  $Rel\ Rel'$

**where**  $B1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $B2: SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$   
**and**  $B3: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $B4: \forall (P, Q) \in Rel'. P \in ProcS \longleftrightarrow Q \in ProcT$   
**and**  $B5: trans (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel')$

**by** *blast*

**have** *fully-abstract*  $SRel\ TRel$

**proof** *auto*

**fix**  $S1\ S2$

**have**  $(TargetTerm\ (\llbracket S1 \rrbracket), SourceTerm\ S1) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**moreover assume**  $(S1, S2) \in SRel$

**with**  $B2$  **have**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**moreover from**  $B1$

**have**  $(SourceTerm\ S2, TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**ultimately have**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel \cup Rel'$

**using**  $B5$

**unfolding** *trans-def*

**by** *blast*

**with**  $B3\ B4$  **show**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$

**by** *blast*

**next**

**fix**  $S1\ S2$

**from**  $B1$

**have**  $(SourceTerm\ S1, TargetTerm\ (\llbracket S1 \rrbracket)) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**moreover assume**  $(\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel$

**with**  $B3$

**have**  $(TargetTerm\ (\llbracket S1 \rrbracket), TargetTerm\ (\llbracket S2 \rrbracket)) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**moreover**

**have**  $(TargetTerm\ (\llbracket S2 \rrbracket), SourceTerm\ S2) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**ultimately have**  $(SourceTerm\ S1, SourceTerm\ S2) \in Rel \cup Rel'$

**using**  $B5$

**unfolding** *trans-def*

**by** *blast*

**with**  $B2\ B4$  **show**  $(S1, S2) \in SRel$

**by** *blast*

**qed**

**moreover have** *refl*  $SRel$

**unfolding** *refl-on-def*

**proof** *auto*

**fix**  $S$

**from**  $B1$  **have**  $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**moreover**

**have**  $(TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} \cup Rel'$

**by** *simp*

**ultimately have**  $(SourceTerm\ S, SourceTerm\ S) \in Rel \cup Rel'$

**using**  $B5$

**unfolding** *trans-def*

**by** *blast*

**with**  $B2\ B4$  **show**  $(S, S) \in SRel$

```

    by blast
qed
moreover have trans TRel
  unfolding trans-def
proof clarify
  fix TP TQ TR
  assume (TP, TQ) ∈ TRel and (TQ, TR) ∈ TRel
  with B3 B4 B5 show (TP, TR) ∈ TRel
    unfolding trans-def
    by blast
qed
ultimately show fully-abstract SRel TRel ∧ refl SRel ∧ trans TRel
  by blast
qed

end
theory CombinedCriteria
  imports DivergenceReflection SuccessSensitiveness FullAbstraction OperationalCorrespondence
begin

```

## 10 Combining Criteria

So far we considered the effect of single criteria on encodings. Often the quality of an encoding is prescribed by a set of different criteria. In the following we analyse the combined effect of criteria. This way we can compare criteria as well as identify side effects that result from combinations of criteria. We start with some technical lemmata. To combine the effect of different criteria we combine the conditions they induce. If their effect can be described by a predicate on the pairs of the relation, as in the case of success sensitiveness or divergence reflection, combining the effects is simple.

**lemma** (in *encoding*) *criterion-iff-source-target-relation-impl-indRelR*:

```

fixes Cond :: ('procS ⇒ 'procT) ⇒ bool
  and Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set ⇒ bool
assumes Cond enc = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ Pred Rel)
shows Cond enc = (∃ Rel'. Pred (indRelR ∪ Rel'))
proof (rule iffI)
  assume Cond enc
  with assms obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel and A2: Pred Rel
    by blast
  from A1 have Rel = indRelR ∪ (Rel - indRelR)
    by (auto simp add: indRelR.simps)
  with A2 have Pred (indRelR ∪ (Rel - indRelR))
    by simp
  thus ∃ Rel'. Pred (indRelR ∪ Rel')
    by blast
next
  assume ∃ Rel'. Pred (indRelR ∪ Rel')
  from this obtain Rel' where Pred (indRelR ∪ Rel')
    by blast
  moreover have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ (indRelR ∪ Rel')
    by (simp add: indRelR.encR)
  ultimately show Cond enc
    using assms
    by blast
qed

```

**lemma** (in *encoding*) *combine-conditions-on-pairs-of-relations*:

```

fixes RelA RelB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  and CondA CondB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
assumes ∀ (P, Q) ∈ RelA. CondA (P, Q)
  and ∀ (P, Q) ∈ RelB. CondB (P, Q)

```

**shows**  $(\forall (P, Q) \in RelA \cap RelB. CondA (P, Q)) \wedge (\forall (P, Q) \in RelA \cap RelB. CondB (P, Q))$   
**using** *assms*  
**by** *blast*

**lemma** (in *encoding*) *combine-conditions-on-sets-of-relations*:

**fixes** *Rel RelA* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set$   
**and** *Cond* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set \Rightarrow bool$   
**and** *CondA* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) \Rightarrow bool$   
**assumes**  $\forall (P, Q) \in RelA. CondA (P, Q)$   
**and**  $Cond Rel \wedge Rel \subseteq RelA$   
**shows**  $Cond Rel \wedge (\forall (P, Q) \in Rel. CondA (P, Q))$   
**using** *assms*  
**by** *blast*

**lemma** (in *encoding*) *combine-conditions-on-sets-and-pairs-of-relations*:

**fixes** *Rel RelA RelB* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set$   
**and** *Cond* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set \Rightarrow bool$   
**and** *CondA CondB* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) \Rightarrow bool$   
**assumes**  $\forall (P, Q) \in RelA. CondA (P, Q)$   
**and**  $\forall (P, Q) \in RelB. CondB (P, Q)$   
**and**  $Cond Rel \wedge Rel \subseteq RelA \wedge Rel \subseteq RelB$   
**shows**  $Cond Rel \wedge (\forall (P, Q) \in Rel. CondA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q))$   
**using** *assms*  
**by** *blast*

We mapped several criteria on conditions on relations that relate at least all source terms and their literal translations. The following lemmata help us to combine such conditions by switching to the witness *indRelR*.

**lemma** (in *encoding*) *combine-conditions-on-relations-indRelR*:

**fixes** *RelA RelB* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set$   
**and** *Cond* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) set \Rightarrow bool$   
**and** *CondA CondB* ::  $((\text{'procS}, \text{'procT}) Proc \times (\text{'procS}, \text{'procT}) Proc) \Rightarrow bool$   
**assumes** *A1*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in RelA$   
**and** *A2*:  $\forall (P, Q) \in RelA. CondA (P, Q)$   
**and** *A3*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in RelB$   
**and** *A4*:  $\forall (P, Q) \in RelB. CondB (P, Q)$   
**shows**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge (\forall (P, Q) \in Rel. CondA (P, Q))$   
 $\wedge (\forall (P, Q) \in Rel. CondB (P, Q))$   
**and**  $Cond indRelR \Longrightarrow (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall (P, Q) \in Rel. CondA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q)) \wedge Cond Rel)$

**proof** –

**have** *A5*:  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in indRelR$   
**by** (*simp add: indRelR.encR*)

**moreover have** *A6*:  $indRelR \subseteq RelA$

**proof** *clarify*

**fix** *P Q*

**assume**  $(P, Q) \in indRelR$

**from this A1 show**  $(P, Q) \in RelA$

**by** (*induct, simp*)

**qed**

**moreover have** *A7*:  $indRelR \subseteq RelB$

**proof** *clarify*

**fix** *P Q*

**assume**  $(P, Q) \in indRelR$

**from this A3 show**  $(P, Q) \in RelB$

**by** (*induct, simp*)

**qed**

**ultimately show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$

$\wedge (\forall (P, Q) \in Rel. CondA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q))$

**using** *combine-conditions-on-sets-and-pairs-of-relations* [where *RelA=RelA* and *RelB=RelB*]

**and**  $CondA=CondA$  **and**  $CondB=CondB$  **and**  $Rel=indRelR$   
**and**  $Cond=\lambda R. \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in R]$   $A2\ A4$   
**by** *blast*  
**from**  $A2\ A4\ A5\ A6\ A7$   
**show**  $Cond\ indRelR \implies (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall (P, Q) \in Rel. CondA\ (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB\ (P, Q)) \wedge Cond\ Rel)$   
**using** *combine-conditions-on-sets-and-pairs-of-relations*[**where**  $RelA=RelA$  **and**  $RelB=RelB$   
**and**  $CondA=CondA$  **and**  $CondB=CondB$  **and**  $Rel=indRelR$   
**and**  $Cond=\lambda R. \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in R \wedge Cond\ R]$   
**by** *blast*  
**qed**

**lemma** (*in encoding*) *indRelR-cond-respects-predA-and-reflects-predB*:  
**fixes**  $PredA\ PredB :: ('procS, 'procT)\ Proc \Rightarrow bool$   
**shows**  $((\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA)$   
 $\wedge (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-pred\ Rel\ PredB))$   
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA$   
 $\wedge rel-reflects-pred\ Rel\ PredB)$   
**proof** (*rule iffI, erule conjE*)  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA$   
**from** *this* **obtain**  $RelA$  **where**  $A1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in RelA$   
**and**  $A2: rel-respects-pred\ RelA\ PredA$   
**by** *blast*  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-pred\ Rel\ PredB$   
**from** *this* **obtain**  $RelB$  **where**  $A3: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in RelB$   
**and**  $A4: rel-reflects-pred\ RelB\ PredB$   
**by** *blast*  
**from**  $A2$  **have**  $\forall (P, Q) \in RelA. PredA\ P \longleftrightarrow PredA\ Q$   
**by** *blast*  
**moreover** **from**  $A4$  **have**  $\forall (P, Q) \in RelB. PredB\ Q \longrightarrow PredB\ P$   
**by** *blast*  
**ultimately** **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall (P, Q) \in Rel. PredA\ P = PredA\ Q) \wedge (\forall (P, Q) \in Rel. PredB\ Q \longrightarrow PredB\ P)$   
**using** *combine-conditions-on-relations-indRelR(1)*[**where**  $RelA=RelA$  **and**  $RelB=RelB$  **and**  
 $CondA=\lambda(P, Q). PredA\ P \longleftrightarrow PredA\ Q$  **and**  $CondB=\lambda(P, Q). PredB\ Q \longrightarrow PredB\ P]$   $A1\ A3$   
**by** *simp*  
**thus**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA$   
 $\wedge rel-reflects-pred\ Rel\ PredB$   
**by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA$   
 $\wedge rel-reflects-pred\ Rel\ PredB$   
**thus**  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA) \wedge$   
 $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-pred\ Rel\ PredB)$   
**by** *blast*  
**qed**

## 10.1 Divergence Reflection and Success Sensitiveness

We combine results on divergence reflection and success sensitiveness to analyse their combined effect on an encoding function. An encoding is success sensitive and reflects divergence iff there exists a relation that relates source terms and their literal translations that reflects divergence and respects success.

**lemma** (*in encoding-wrt-barbs*) *WSS-DR-iff-source-target-rel*:  
**fixes**  $success :: 'barbs$   
**shows** (*enc-weakly-respects-barb-set*  $\{success\} \wedge enc-reflects-divergence$ )  
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target))$   
**proof** –

**have**  $\forall Rel. \text{rel-reflects-divergence } Rel \ (STCal \ Source \ Target)$   
 $= \text{rel-reflects-pred } Rel \ \text{divergentST}$   
**by** (*simp add: divergentST-STCal-divergent*)  
**moreover have**  $\forall Rel. (\text{rel-weakly-respects-barb-set } Rel \ (STCalWB \ SWB \ TWB) \ \{success\})$   
 $= \text{rel-respects-pred } Rel \ (\lambda P. P \downarrow . \text{success})$   
**by** (*simp add: STCalWB-reachesBarbST*)  
**ultimately show**  $(\text{enc-weakly-respects-barb-set } \{success\} \wedge \text{enc-reflects-divergence})$   
 $= (\exists Rel. (\forall S. (SourceTerm \ S, \ TargetTerm \ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-weakly-respects-barb-set } Rel \ (STCalWB \ SWB \ TWB) \ \{success\})$   
 $\wedge \text{rel-reflects-divergence } Rel \ (STCal \ Source \ Target))$   
**using** *success-sensitive-iff-source-target-rel-weakly-respects-success(1)*  
*divergence-reflection-iff-source-target-rel-reflects-divergence*  
*indRelR-cond-respects-predA-and-reflects-predB* [**where**  
 $PredA = \lambda P. P \downarrow . \text{success}$  **and**  $PredB = \text{divergentST}$ ]  
**by** *simp*  
**qed**

**lemma** (*in encoding-wrt-barbs*) *SS-DR-iff-source-target-rel:*  
**fixes** *success* :: 'barbs  
**shows**  $(\text{enc-respects-barb-set } \{success\} \wedge \text{enc-reflects-divergence})$   
 $= (\exists Rel. (\forall S. (SourceTerm \ S, \ TargetTerm \ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-respects-barb-set } Rel \ (STCalWB \ SWB \ TWB) \ \{success\})$   
 $\wedge \text{rel-reflects-divergence } Rel \ (STCal \ Source \ Target))$

**proof** –  
**have**  $\forall Rel. \text{rel-reflects-divergence } Rel \ (STCal \ Source \ Target)$   
 $= \text{rel-reflects-pred } Rel \ \text{divergentST}$   
**by** (*simp add: divergentST-STCal-divergent*)  
**moreover have**  $\forall Rel. (\text{rel-respects-barb-set } Rel \ (STCalWB \ SWB \ TWB) \ \{success\})$   
 $= \text{rel-respects-pred } Rel \ (\lambda P. P \downarrow . \text{success})$   
**by** (*simp add: STCalWB-hasBarbST*)  
**ultimately show**  $(\text{enc-respects-barb-set } \{success\} \wedge \text{enc-reflects-divergence})$   
 $= (\exists Rel. (\forall S. (SourceTerm \ S, \ TargetTerm \ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-respects-barb-set } Rel \ (STCalWB \ SWB \ TWB) \ \{success\})$   
 $\wedge \text{rel-reflects-divergence } Rel \ (STCal \ Source \ Target))$   
**using** *success-sensitive-iff-source-target-rel-respects-success(1)*  
*divergence-reflection-iff-source-target-rel-reflects-divergence*  
*indRelR-cond-respects-predA-and-reflects-predB* [**where**  
 $PredA = \lambda P. P \downarrow . \text{success}$  **and**  $PredB = \text{divergentST}$ ]  
**by** *simp*  
**qed**

## 10.2 Adding Operational Correspondence

The effect of operational correspondence includes conditions (TRel is included, transitivity) that require a witness like indRelRTPO. In order to combine operational correspondence with success sensitivity, we show that if the encoding and TRel (weakly) respects barbs than indRelRTPO (weakly) respects barbs. Since success is only a specific kind of barbs, the same holds for success sensitivity.

**lemma** (*in encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-weakly-respects-success:*  
**fixes** *success* :: 'barbs  
**and** *TRel* :: ('procT  $\times$  'procT) set  
**assumes** *encRS*: *enc-weakly-respects-barb-set* {*success*}  
**and** *trelPS*: *rel-weakly-preserves-barb-set* *TRel* *TWB* {*success*}  
**and** *trelRS*: *rel-weakly-reflects-barb-set* *TRel* *TWB* {*success*}  
**shows** *rel-weakly-respects-barb-set* (*indRelRTPO* *TRel*) (*STCalWB* *SWB* *TWB*) {*success*}  
**proof** *auto*  
**fix** *P Q P'*  
**assume**  $P \lesssim \llbracket \cdot \rrbracket RT < TRel > Q$  **and**  $P \mapsto (\text{Calculus } (STCalWB \ SWB \ TWB)) * P'$   
**and**  $P' \downarrow < STCalWB \ SWB \ TWB > \text{success}$   
**thus**  $Q \downarrow < STCalWB \ SWB \ TWB > \text{success}$   
**proof** (*induct arbitrary: P'*)

```

case (encR S)
assume SourceTerm S  $\mapsto$  (Calculus (STCalWB SWB TWB))* P' and  $P' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
hence  $S \Downarrow \langle SWB \rangle \text{success}$ 
  using STCalWB-reachesBarbST
  by blast
with encRS have  $\llbracket S \rrbracket \Downarrow \langle TWB \rangle \text{success}$ 
  by simp
thus TargetTerm ( $\llbracket S \rrbracket$ )  $\Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  using STCalWB-reachesBarbST
  by blast
next
case (source S)
assume SourceTerm S  $\mapsto$  (Calculus (STCalWB SWB TWB))* P' and  $P' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
thus SourceTerm S  $\Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  by blast
next
case (target T1 T2)
assume (T1, T2)  $\in$  TRel
moreover assume TargetTerm T1  $\mapsto$  (Calculus (STCalWB SWB TWB))* P'
  and  $P' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
hence  $T1 \Downarrow \langle TWB \rangle \text{success}$ 
  using STCalWB-reachesBarbST
  by blast
ultimately have  $T2 \Downarrow \langle TWB \rangle \text{success}$ 
  using trelPS
  by simp
thus TargetTerm T2  $\Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  using STCalWB-reachesBarbST
  by blast
next
case (trans P Q R)
assume P  $\mapsto$  (Calculus (STCalWB SWB TWB))* P' and  $P' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  and  $\bigwedge P'. P \mapsto (\text{Calculus (STCalWB SWB TWB)}) * P' \implies P' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
   $\implies Q \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
hence  $Q \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  by simp
moreover assume  $\bigwedge Q'. Q \mapsto (\text{Calculus (STCalWB SWB TWB)}) * Q' \implies Q' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
   $\implies R \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
ultimately show  $R \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  by blast
qed
next
fix P Q Q'
assume  $P \lesssim \llbracket \cdot \rrbracket RT \langle TRel \rangle Q$  and  $Q \mapsto (\text{Calculus (STCalWB SWB TWB)}) * Q'$ 
  and  $Q' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
thus  $P \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
proof (induct arbitrary: Q')
  case (encR S)
  assume TargetTerm ( $\llbracket S \rrbracket$ )  $\mapsto$  (Calculus (STCalWB SWB TWB))* Q'
    and  $Q' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
  hence  $\llbracket S \rrbracket \Downarrow \langle TWB \rangle \text{success}$ 
    using STCalWB-reachesBarbST
    by blast
  with encRS have  $S \Downarrow \langle SWB \rangle \text{success}$ 
    by simp
  thus SourceTerm S  $\Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 
    using STCalWB-reachesBarbST
    by blast
next
case (source S)
assume SourceTerm S  $\mapsto$  (Calculus (STCalWB SWB TWB))* Q' and  $Q' \Downarrow \langle STCalWB SWB TWB \rangle \text{success}$ 

```

```

thus SourceTerm  $S \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
  by blast
next
  case (target  $T1\ T2$ )
  assume  $(T1, T2) \in TRel$ 
  moreover assume  $TargetTerm\ T2 \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* Q'$ 
    and  $Q' \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
  hence  $T2 \Downarrow \langle TWB \rangle success$ 
    using STCalWB-reachesBarbST
    by blast
  ultimately have  $T1 \Downarrow \langle TWB \rangle success$ 
    using trelRS
    by blast
thus TargetTerm  $T1 \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
  using STCalWB-reachesBarbST
  by blast
next
  case (trans  $P\ Q\ R\ R'$ )
  assume  $R \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* R'$  and  $R' \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
    and  $\bigwedge R'. R \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* R' \implies R' \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
     $\implies Q \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
  hence  $Q \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
    by simp
  moreover assume  $\bigwedge Q'. Q \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* Q' \implies Q' \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
     $\implies P \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
  ultimately show  $P \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$ 
    by blast
qed

```

**lemma** (*in encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs*:

```

fixes TRel :: ('procT × 'procT) set
assumes encRS: enc-weakly-respects-barbs
  and trelPS: rel-weakly-preserves-barbs TRel TWB
  and trelRS: rel-weakly-reflects-barbs TRel TWB
shows rel-weakly-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
proof auto
fix  $P\ Q\ x\ P'$ 
assume  $P \lesssim [\cdot] RT \langle TRel \rangle Q$  and  $P \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* P'$ 
  and  $P' \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
thus  $Q \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
proof (induct arbitrary: P')
  case (encR  $S$ )
  assume  $SourceTerm\ S \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* P'$  and  $P' \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
  hence  $S \Downarrow \langle SWB \rangle x$ 
    using STCalWB-reachesBarbST
    by blast
  with encRS have  $\llbracket S \rrbracket \Downarrow \langle TWB \rangle x$ 
    by simp
  thus TargetTerm  $(\llbracket S \rrbracket) \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
    using STCalWB-reachesBarbST
    by blast
next
  case (source  $S$ )
  assume  $SourceTerm\ S \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* P'$  and  $P' \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
  thus  $SourceTerm\ S \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$ 
    by blast
next
  case (target  $T1\ T2$ )
  assume  $(T1, T2) \in TRel$ 
  moreover assume  $TargetTerm\ T1 \mapsto (Calculus\ (STCalWB\ SWB\ TWB))^* P'$ 

```



**and**  $P' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**hence**  $T1 \Downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $T2 \Downarrow \langle TWB \rangle x$   
**using** *trelPS*  
**by** *simp*  
**thus** *TargetTerm*  $T2 \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**case** (*trans P Q R*)  
**assume**  $P \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * P'$  **and**  $P' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**and**  $\bigwedge P'. P \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * P' \implies P' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
 $\implies Q \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**hence**  $Q \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**by** *simp*  
**moreover assume**  $\bigwedge Q'. Q \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * Q' \implies Q' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
 $\implies R \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**ultimately show**  $R \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**by** *blast*  
**qed**  
**next**  
**fix**  $P \ Q \ x \ Q'$   
**assume**  $P \lesssim [\cdot] RT \langle TRel \rangle Q$  **and**  $Q \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * Q'$   
**and**  $Q' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**thus**  $P \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**proof** (*induct arbitrary: Q'*)  
**case** (*encR S*)  
**assume** *TargetTerm*  $([S]) \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * Q'$   
**and**  $Q' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**hence**  $[S] \Downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**with** *encRS* **have**  $S \Downarrow \langle SWB \rangle x$   
**by** *simp*  
**thus** *SourceTerm*  $S \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**case** (*source S*)  
**assume** *SourceTerm*  $S \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * Q'$  **and**  $Q' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**thus** *SourceTerm*  $S \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**by** *blast*  
**next**  
**case** (*target T1 T2*)  
**assume**  $(T1, T2) \in TRel$   
**moreover assume** *TargetTerm*  $T2 \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * Q'$   
**and**  $Q' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**hence**  $T2 \Downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $T1 \Downarrow \langle TWB \rangle x$   
**using** *trelRS*  
**by** *blast*  
**thus** *TargetTerm*  $T1 \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**case** (*trans P Q R R'*)  
**assume**  $R \mapsto (Calculus \ (STCalWB \ SWB \ TWB)) * R'$  **and**  $R' \Downarrow \langle STCalWB \ SWB \ TWB \rangle x$

**and**  $\bigwedge R'. R \mapsto (\text{Calculus } (STCalWB \ SWB \ TWB)) * R' \implies R' \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
 $\implies Q \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**hence**  $Q \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**by** *simp*  
**moreover assume**  $\bigwedge Q'. Q \mapsto (\text{Calculus } (STCalWB \ SWB \ TWB)) * Q' \implies Q' \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
 $\implies P \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**ultimately show**  $P \downarrow \langle STCalWB \ SWB \ TWB \rangle x$   
**by** *blast*  
**qed**  
**qed**

**lemma** (in *encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-respects-success*:

**fixes** *success* :: 'barbs  
**and** *TRel* :: ('procT × 'procT) set  
**assumes** *encRS*: *enc-respects-barb-set* {*success*}  
**and** *trelPS*: *rel-preserves-barb-set* *TRel* *TWB* {*success*}  
**and** *trelRS*: *rel-reflects-barb-set* *TRel* *TWB* {*success*}  
**shows** *rel-respects-barb-set* (*indRelRTPO* *TRel*) (*STCalWB* *SWB* *TWB*) {*success*}  
**proof** *auto*  
**fix** *P* *Q*  
**assume**  $P \lesssim [\cdot] RT \langle TRel \rangle Q$  **and**  $P \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**thus**  $Q \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**proof** *induct*  
**case** (*encR* *S*)  
**assume** *SourceTerm*  $S \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**hence**  $S \downarrow \langle SWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**with** *encRS* **have**  $\llbracket S \rrbracket \downarrow \langle TWB \rangle success$   
**by** *simp*  
**thus** *TargetTerm*  $(\llbracket S \rrbracket) \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**next**  
**case** (*source* *S*)  
**assume** *SourceTerm*  $S \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**thus** *SourceTerm*  $S \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**by** *simp*  
**next**  
**case** (*target* *T1* *T2*)  
**assume**  $(T1, T2) \in TRel$   
**moreover assume** *TargetTerm*  $T1 \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**hence**  $T1 \downarrow \langle TWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**ultimately have**  $T2 \downarrow \langle TWB \rangle success$   
**using** *trelPS*  
**by** *simp*  
**thus** *TargetTerm*  $T2 \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**next**  
**case** (*trans* *P* *Q* *R*)  
**assume**  $P \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**and**  $P \downarrow \langle STCalWB \ SWB \ TWB \rangle success \implies Q \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**and**  $Q \downarrow \langle STCalWB \ SWB \ TWB \rangle success \implies R \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**thus**  $R \downarrow \langle STCalWB \ SWB \ TWB \rangle success$   
**by** *simp*  
**qed**  
**next**  
**fix** *P* *Q*

```

assume  $P \lesssim \llbracket \cdot \rrbracket RT < TRel > Q$  and  $Q \downarrow < STCalWB SWB TWB > success$ 
thus  $P \downarrow < STCalWB SWB TWB > success$ 
proof induct
  case (encR S)
    assume TargetTerm  $\llbracket S \rrbracket \downarrow < STCalWB SWB TWB > success$ 
    hence  $\llbracket S \rrbracket \downarrow < TWB > success$ 
      using STCalWB-hasBarbST
      by blast
    with encRS have  $S \downarrow < SWB > success$ 
      by simp
    thus SourceTerm  $S \downarrow < STCalWB SWB TWB > success$ 
      using STCalWB-hasBarbST
      by blast
  next
    case (source S)
      assume SourceTerm  $S \downarrow < STCalWB SWB TWB > success$ 
      thus SourceTerm  $S \downarrow < STCalWB SWB TWB > success$ 
        by simp
  next
    case (target T1 T2)
      assume  $(T1, T2) \in TRel$ 
      moreover assume TargetTerm  $T2 \downarrow < STCalWB SWB TWB > success$ 
      hence  $T2 \downarrow < TWB > success$ 
        using STCalWB-hasBarbST
        by blast
      ultimately have  $T1 \downarrow < TWB > success$ 
        using treIRS
        by blast
      thus TargetTerm  $T1 \downarrow < STCalWB SWB TWB > success$ 
        using STCalWB-hasBarbST
        by blast
  next
    case (trans P Q R)
      assume  $R \downarrow < STCalWB SWB TWB > success$ 
        and  $R \downarrow < STCalWB SWB TWB > success \implies Q \downarrow < STCalWB SWB TWB > success$ 
        and  $Q \downarrow < STCalWB SWB TWB > success \implies P \downarrow < STCalWB SWB TWB > success$ 
      thus  $P \downarrow < STCalWB SWB TWB > success$ 
        by simp
qed
qed

```

**lemma** (*in encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-respects-barbs*:

```

fixes TRel :: ('procT × 'procT) set
assumes encRS: enc-respects-barbs
  and treIPS: rel-preserves-barbs TRel TWB
  and treIRS: rel-reflects-barbs TRel TWB
shows rel-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
proof auto
fix  $P Q x$ 
assume  $P \lesssim \llbracket \cdot \rrbracket RT < TRel > Q$  and  $P \downarrow < STCalWB SWB TWB > x$ 
thus  $Q \downarrow < STCalWB SWB TWB > x$ 
proof induct
  case (encR S)
    assume SourceTerm  $S \downarrow < STCalWB SWB TWB > x$ 
    hence  $S \downarrow < SWB > x$ 
      using STCalWB-hasBarbST
      by blast
    with encRS have  $\llbracket S \rrbracket \downarrow < TWB > x$ 
      by simp
    thus TargetTerm  $\llbracket S \rrbracket \downarrow < STCalWB SWB TWB > x$ 
      using STCalWB-hasBarbST

```

```

    by blast
next
  case (source S)
  assume SourceTerm S↓<STCalWB SWB TWB>x
  thus SourceTerm S↓<STCalWB SWB TWB>x
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  moreover assume TargetTerm T1↓<STCalWB SWB TWB>x
  hence T1↓<TWB>x
    using STCalWB-hasBarbST
    by blast
  ultimately have T2↓<TWB>x
    using trelPS
    by simp
  thus TargetTerm T2↓<STCalWB SWB TWB>x
    using STCalWB-hasBarbST
    by blast
next
  case (trans P Q R)
  assume P↓<STCalWB SWB TWB>x
    and P↓<STCalWB SWB TWB>x ⇒ Q↓<STCalWB SWB TWB>x
    and Q↓<STCalWB SWB TWB>x ⇒ R↓<STCalWB SWB TWB>x
  thus R↓<STCalWB SWB TWB>x
    by simp
qed
next
  fix P Q x
  assume P ≲[·]RT<TRel> Q and Q↓<STCalWB SWB TWB>x
  thus P↓<STCalWB SWB TWB>x
  proof induct
    case (encR S)
    assume TargetTerm ([S])↓<STCalWB SWB TWB>x
    hence [S]↓<TWB>x
      using STCalWB-hasBarbST
      by blast
    with encRS have S↓<SWB>x
      by simp
    thus SourceTerm S↓<STCalWB SWB TWB>x
      using STCalWB-hasBarbST
      by blast
  next
    case (source S)
    assume SourceTerm S↓<STCalWB SWB TWB>x
    thus SourceTerm S↓<STCalWB SWB TWB>x
      by simp
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    moreover assume TargetTerm T2↓<STCalWB SWB TWB>x
    hence T2↓<TWB>x
      using STCalWB-hasBarbST
      by blast
    ultimately have T1↓<TWB>x
      using trelRS
      by blast
  thus TargetTerm T1↓<STCalWB SWB TWB>x
    using STCalWB-hasBarbST
    by blast
next

```

```

case (trans P Q R)
assume  $R \downarrow \langle STCalWB SWB TWB \rangle x$ 
  and  $R \downarrow \langle STCalWB SWB TWB \rangle x \implies Q \downarrow \langle STCalWB SWB TWB \rangle x$ 
  and  $Q \downarrow \langle STCalWB SWB TWB \rangle x \implies P \downarrow \langle STCalWB SWB TWB \rangle x$ 
thus  $P \downarrow \langle STCalWB SWB TWB \rangle x$ 
  by simp
qed
qed

```

An encoding is success sensitive and operational corresponding w.r.t. a bisimulation  $TRel$  that respects success iff there exists a bisimulation that includes  $TRel$  and respects success. The same holds if we consider not only success sensitiveness but barb sensitiveness in general.

**lemma** (*in encoding-wrt-barbs*) *OC-SS-iff-source-target-rel*:

```

fixes success :: 'barbs
and  $TRel$  :: ('procT × 'procT) set
shows (operational-corresponding (TRel*))
  ∧ weak-reduction-bisimulation (TRel+) Target
  ∧ enc-weakly-respects-barb-set {success}
  ∧ rel-weakly-respects-barb-set TRel TWB {success})
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ )) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈  $TRel \longrightarrow$  (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel  $\longrightarrow$  (T1, T2) ∈  $TRel^+$ )
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T) ∈  $TRel^*$ )
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
  ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success})
proof (rule iffI, (erule conjE)+)
assume A1: rel-weakly-preserves-barb-set TRel TWB {success}
and A2: rel-weakly-reflects-barb-set TRel TWB {success}
and A3: enc-weakly-preserves-barb-set {success}
and A4: enc-weakly-reflects-barb-set {success}
define Rel where  $Rel = indRelRTPO TRel$ 
hence B1: ∀ S. (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ )) ∈ Rel
  by (simp add: indRelRTPO.encR)
from Rel-def have B2: ∀ T1 T2. (T1, T2) ∈ TRel  $\longrightarrow$  (TargetTerm T1, TargetTerm T2) ∈ Rel
  by (simp add: indRelRTPO.target)
from Rel-def have B3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel  $\longrightarrow$  (T1, T2) ∈ TRel+
  by (simp add: indRelRTPO-to-TRel(4)[where TRel=TRel])
from Rel-def have B4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T) ∈ TRel*
  using indRelRTPO-to-TRel(2)[where TRel=TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  by simp
assume operational-complete (TRel*)
and operational-sound (TRel*)
and weak-reduction-simulation (TRel+) Target
and  $\forall P Q Q'. (P, Q) \in TRel^+ \wedge Q \mapsto Target^* Q'$ 
   $\longrightarrow (\exists P'. P \mapsto Target^* P' \wedge (P', Q') \in TRel^+)$ 
with Rel-def have B5: weak-reduction-bisimulation Rel (STCal Source Target)
  using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel=TRel]
  by simp
from Rel-def A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel]
  and success=success]
  by blast
show  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$ 
   $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel)$ 
   $\wedge (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$ 
   $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$ 
   $\wedge$  weak-reduction-bisimulation Rel (STCal Source Target)
   $\wedge$  rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
apply (rule exI) using B1 B2 B3 B4 B5 B6 by blast

```

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge \text{weak-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
 $\wedge \text{rel-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$

**from this obtain Rel where**  $C1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $C2: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**and**  $C3: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**and**  $C4: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**and**  $C5: \text{weak-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
**and**  $C6: \text{rel-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**by auto**

**from**  $C1\ C2\ C3\ C4\ C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge \text{weak-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
**by blast**

**hence operational-corresponding**  $(TRel^*)$   
 $\wedge \text{weak-reduction-bisimulation}\ (TRel^+)\ Target$   
**using**  $OC\text{-iff-weak-reduction-bisimulation}[\text{where } TRel=TRel]$   
**by auto**

**moreover have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**apply**  $(rule\ exI)$  **using**  $C1\ C6$  **by blast**

**hence enc-weakly-respects-barb-set**  $\{success\}$   
**using**  $success\text{-sensitive-iff-source-target-rel-weakly-respects-success}$   
**by auto**

**moreover have**  $\text{rel-weakly-respects-barb-set}\ TRel\ TWB\ \{success\}$

**proof auto**  
**fix**  $TP\ TQ\ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by simp**  
**moreover assume**  $TP \mapsto (\text{Calculus}\ TWB)^* TP'$  **and**  $TP' \Downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $STCalWB\text{-reachesBarbST}$   
**by blast**  
**ultimately have**  $TargetTerm\ TQ \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by blast**  
**thus**  $TQ \Downarrow \langle TWB \rangle success$   
**using**  $STCalWB\text{-reachesBarbST}$   
**by blast**

**next**  
**fix**  $TP\ TQ\ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by simp**  
**moreover assume**  $TQ \mapsto (\text{Calculus}\ TWB)^* TQ'$  **and**  $TQ' \Downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TQ \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $STCalWB\text{-reachesBarbST}$   
**by blast**  
**ultimately have**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by blast**  
**thus**  $TP \Downarrow \langle TWB \rangle success$   
**using**  $STCalWB\text{-reachesBarbST}$   
**by blast**

qed  
ultimately show *operational-corresponding* ( $TRel^*$ )  
 $\wedge$  *weak-reduction-bisimulation* ( $TRel^+$ ) *Target*  
 $\wedge$  *enc-weakly-respects-barb-set* {*success*}  $\wedge$  *rel-weakly-respects-barb-set*  $TRel$   $TWB$  {*success*}  
by *fast*  
qed

lemma (in *encoding-wrt-barbs*) *OC-SS-RB-iff-source-target-rel*:

fixes *success* :: 'barbs  
and  $TRel$  :: ('procT  $\times$  'procT) set  
shows (*operational-corresponding* ( $TRel^*$ )  
 $\wedge$  *weak-reduction-bisimulation* ( $TRel^+$ ) *Target*  
 $\wedge$  *enc-weakly-respects-barbs*  $\wedge$  *enc-weakly-respects-barb-set* {*success*}  
 $\wedge$  *rel-weakly-respects-barbs*  $TRel$   $TWB$   $\wedge$  *rel-weakly-respects-barb-set*  $TRel$   $TWB$  {*success*})  
= ( $\exists Rel$ . ( $\forall S$ . (*SourceTerm*  $S$ , *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$   $Rel$ )  
 $\wedge$  ( $\forall T1$   $T2$ . ( $T1$ ,  $T2$ )  $\in$   $TRel$   $\longrightarrow$  (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$ )  
 $\wedge$  ( $\forall T1$   $T2$ . (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$   $\longrightarrow$  ( $T1$ ,  $T2$ )  $\in$   $TRel^+$ )  
 $\wedge$  ( $\forall S$   $T$ . (*SourceTerm*  $S$ , *TargetTerm*  $T$ )  $\in$   $Rel$   $\longrightarrow$  ( $\llbracket S \rrbracket$ ,  $T$ )  $\in$   $TRel^*$ )  
 $\wedge$  *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)  
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB*  $TWB$ )  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB*  $TWB$ ) {*success*})

proof (rule *iffI*, (*erule conjE*) $+$ )

assume  $A1$ : *rel-weakly-preserves-barb-set*  $TRel$   $TWB$  {*success*}  
and  $A2$ : *rel-weakly-reflects-barb-set*  $TRel$   $TWB$  {*success*}  
and  $A3$ : *enc-weakly-preserves-barb-set* {*success*}  
and  $A4$ : *enc-weakly-reflects-barb-set* {*success*}  
and  $A5$ : *rel-weakly-preserves-barbs*  $TRel$   $TWB$  and  $A6$ : *rel-weakly-reflects-barbs*  $TRel$   $TWB$   
and  $A7$ : *enc-weakly-preserves-barbs* and  $A8$ : *enc-weakly-reflects-barbs*

define  $Rel$  where  $Rel = indRelRTPO$   $TRel$

hence  $B1$ :  $\forall S$ . (*SourceTerm*  $S$ , *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$   $Rel$

by (*simp add*: *indRelRTPO.encR*)

from  $Rel$ -def have  $B2$ :  $\forall T1$   $T2$ . ( $T1$ ,  $T2$ )  $\in$   $TRel$   $\longrightarrow$  (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$

by (*simp add*: *indRelRTPO.target*)

from  $Rel$ -def have  $B3$ :  $\forall T1$   $T2$ . (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$   $\longrightarrow$  ( $T1$ ,  $T2$ )  $\in$   $TRel^+$

by (*simp add*: *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ])

from  $Rel$ -def have  $B4$ :  $\forall S$   $T$ . (*SourceTerm*  $S$ , *TargetTerm*  $T$ )  $\in$   $Rel$   $\longrightarrow$  ( $\llbracket S \rrbracket$ ,  $T$ )  $\in$   $TRel^*$

using *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ]

*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]

by *simp*

assume *operational-complete* ( $TRel^*$ )

and *operational-sound* ( $TRel^*$ )

and *weak-reduction-simulation* ( $TRel^+$ ) *Target*

and  $\forall P$   $Q$   $Q'$ . ( $P$ ,  $Q$ )  $\in$   $TRel^+ \wedge Q \longmapsto Target^* Q' \longrightarrow (\exists P'$ .  $P \longmapsto Target^* P' \wedge (P', Q') \in TRel^+)$

with  $Rel$ -def have  $B5$ : *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)

using *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]

by *simp*

from  $Rel$ -def  $A1$   $A2$   $A3$   $A4$  have  $B6$ : *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB*  $TWB$ ) {*success*}

using *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*[**where**  $TRel = TRel$ ]

and *success=success*]

by *blast*

from  $Rel$ -def  $A5$   $A6$   $A7$   $A8$  have  $B7$ : *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB*  $TWB$ )

using *enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs*[**where**  $TRel = TRel$ ]

by *blast*

show  $\exists Rel$ . ( $\forall S$ . (*SourceTerm*  $S$ , *TargetTerm* ( $\llbracket S \rrbracket$ ))  $\in$   $Rel$ )

$\wedge$  ( $\forall T1$   $T2$ . ( $T1$ ,  $T2$ )  $\in$   $TRel$   $\longrightarrow$  (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$ )

$\wedge$  ( $\forall T1$   $T2$ . (*TargetTerm*  $T1$ , *TargetTerm*  $T2$ )  $\in$   $Rel$   $\longrightarrow$  ( $T1$ ,  $T2$ )  $\in$   $TRel^+$ )

$\wedge$  ( $\forall S$   $T$ . (*SourceTerm*  $S$ , *TargetTerm*  $T$ )  $\in$   $Rel$   $\longrightarrow$  ( $\llbracket S \rrbracket$ ,  $T$ )  $\in$   $TRel^*$ )

$\wedge$  *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)

$\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB*  $TWB$ )

$\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB*  $TWB$ ) {*success*}

apply (rule *exI*) using  $B1$   $B2$   $B3$   $B4$   $B5$   $B6$   $B7$  by *blast*

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge weak\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
 $\wedge rel\text{-weakly-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
 $\wedge rel\text{-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$

**from this obtain Rel where**  $C: (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge weak\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
 $\wedge rel\text{-weakly-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
 $\wedge rel\text{-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$

**by auto**

**hence**  $C1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**by simp**

**from C have**  $C2: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$   
**by simp**

**from C have**  $C3: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$   
**by simp**

**from C have**  $C4: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*$   
**by simp**

**from C have**  $C5: weak\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
**by simp**

**from C have**  $C7: rel\text{-weakly-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
**apply (rule conjE) apply (erule conjE)+ by blast**

**from C have**  $C6: rel\text{-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**apply (rule conjE) apply (erule conjE)+ by blast**

**from C1 C2 C3 C4 C5 have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$   
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$   
 $\wedge weak\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
**by blast**

**hence operational-corresponding**  $(TRel^*)$   
 $\wedge weak\text{-reduction-bisimulation}\ (TRel^+)\ Target$   
**using**  $OC\text{-iff-weak-reduction-bisimulation}[\mathbf{where}\ TRel=TRel]$   
**by auto**

**moreover have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-weakly-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**apply (rule exI) using C1 C6 by blast**

**hence enc-weakly-respects-barb-set**  $\{success\}$   
**using**  $success\text{-sensitive-iff-source-target-rel-weakly-respects-success}$   
**by auto**

**moreover have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel\text{-weakly-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
**apply (rule exI) using C1 C7 by blast**

**hence enc-weakly-respects-barbs**  
**using**  $enc\text{-weakly-respects-barbs-iff-source-target-rel}$   
**by auto**

**moreover have**  $rel\text{-weakly-respects-barb-set}\ TRel\ TWB\ \{success\}$

**proof auto**

**fix**  $TP\ TQ\ TP'$

**assume**  $(TP, TQ) \in TRel$

**with C2 have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by simp**

**moreover assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \Downarrow \langle TWB \rangle success$

**hence**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $STCalWB\text{-reachesBarbST}$



```

    by blast
ultimately have TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>success
  using C6
  by blast
thus TQ $\Downarrow$ <TWB>success
  using STCalWB-reachesBarbST
  by blast
next
fix TP TQ TQ'
assume (TP, TQ)  $\in$  TRel
with C2 have (TargetTerm TP, TargetTerm TQ)  $\in$  Rel
  by simp
moreover assume TQ  $\mapsto$ (Calculus TWB)* TQ' and TQ' $\Downarrow$ <TWB>success
hence TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>success
  using STCalWB-reachesBarbST
  by blast
ultimately have TargetTerm TP $\Downarrow$ <STCalWB SWB TWB>success
  using C6
  by blast
thus TP $\Downarrow$ <TWB>success
  using STCalWB-reachesBarbST
  by blast
qed
moreover have rel-weakly-respects-barbs TRel TWB
proof auto
fix TP TQ x TP'
assume (TP, TQ)  $\in$  TRel
with C2 have (TargetTerm TP, TargetTerm TQ)  $\in$  Rel
  by simp
moreover assume TP  $\mapsto$ (Calculus TWB)* TP' and TP' $\Downarrow$ <TWB>x
hence TargetTerm TP $\Downarrow$ <STCalWB SWB TWB>x
  using STCalWB-reachesBarbST
  by blast
ultimately have TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>x
  using C7
  by blast
thus TQ $\Downarrow$ <TWB>x
  using STCalWB-reachesBarbST
  by blast
next
fix TP TQ x TQ'
assume (TP, TQ)  $\in$  TRel
with C2 have (TargetTerm TP, TargetTerm TQ)  $\in$  Rel
  by simp
moreover assume TQ  $\mapsto$ (Calculus TWB)* TQ' and TQ' $\Downarrow$ <TWB>x
hence TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>x
  using STCalWB-reachesBarbST
  by blast
ultimately have TargetTerm TP $\Downarrow$ <STCalWB SWB TWB>x
  using C7
  by blast
thus TP $\Downarrow$ <TWB>x
  using STCalWB-reachesBarbST
  by blast
qed
ultimately show operational-corresponding (TRel*)
 $\wedge$  weak-reduction-bisimulation (TRel+) Target
 $\wedge$  enc-weakly-respects-barbs  $\wedge$  enc-weakly-respects-barb-set {success}
 $\wedge$  rel-weakly-respects-barbs TRel TWB  $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
  by fast
qed

```

**lemma** (in *encoding-wrt-barbs*) *OC-SS-wrt-preorder-iff-source-target-rel*:  
**fixes** *success* :: 'barbs  
**and** *TRel* :: ('procT × 'procT) set  
**shows** (*operational-corresponding TRel* ∧ *preorder TRel* ∧ *weak-reduction-bisimulation TRel Target*  
 ∧ *enc-weakly-respects-barb-set {success}*  
 ∧ *rel-weakly-respects-barb-set TRel TWB {success}*)  
 = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *Rel*)  
 ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}  
 ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* → ( $\llbracket S \rrbracket$ , *T*) ∈ *TRel*)  
 ∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*  
 ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*})  
**proof** (*rule iffI*, (*erule conjE*)<sup>+</sup>)  
**assume** *A1*: *rel-weakly-preserves-barb-set TRel TWB {success}*  
**and** *A2*: *rel-weakly-reflects-barb-set TRel TWB {success}*  
**and** *A3*: *enc-weakly-preserves-barb-set {success}*  
**and** *A4*: *enc-weakly-reflects-barb-set {success}*  
**and** *A5*: *preorder TRel*  
**from** *A5* **have** *A6*: *TRel*<sup>+</sup> = *TRel*  
**using** *trancl-id*[of *TRel*] *preorder-on-def*  
**by** *blast*  
**from** *A5* **have** *A7*: *TRel*<sup>\*</sup> = *TRel*  
**using** *reflcl-trancl*[of *TRel*] *trancl-id*[of *TRel*]  
**unfolding** *refl-on-def preorder-on-def*  
**by** *auto*  
**define** *Rel* **where** *Rel* = *indRelRTPO TRel*  
**hence** *B1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *Rel*  
**by** (*simp add: indRelRTPO.encR*)  
**from** *Rel-def A6* **have** *B2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}  
**using** *indRelRTPO-to-TRel(4)*[**where** *TRel*=*TRel*]  
**by** (*auto simp add: indRelRTPO.target*)  
**from** *Rel-def A7* **have** *B3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* → ( $\llbracket S \rrbracket$ , *T*) ∈ *TRel*  
**using** *indRelRTPO-to-TRel(2)*[**where** *TRel*=*TRel*]  
*trans-closure-of-TRel-refl-cond*[**where** *TRel*=*TRel*]  
**by** *simp*  
**assume** *operational-complete TRel* **and** *operational-sound TRel*  
**and** *weak-reduction-simulation TRel Target*  
**and** ∀ *P Q Q'*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ↦ *Target*\* *Q'* → (∃ *P'*. *P* ↦ *Target*\* *P'* ∧ (*P'*, *Q'*) ∈ *TRel*)  
**with** *Rel-def A6 A7* **have** *B4*: *weak-reduction-bisimulation Rel* (*STCal Source Target*)  
**using** *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]  
**by** *simp*  
**from** *Rel-def A5* **have** *B5*: *preorder Rel*  
**using** *indRelRTPO-is-preorder*[**where** *TRel*=*TRel*]  
**unfolding** *preorder-on-def*  
**by** *blast*  
**from** *Rel-def A1 A2 A3 A4* **have** *B6*: *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}  
**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*[**where** *TRel*=*TRel*]  
**and** *success=success*  
**by** *blast*  
**show** ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *Rel*)  
 ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}  
 ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* → ( $\llbracket S \rrbracket$ , *T*) ∈ *TRel*)  
 ∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*  
 ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}  
**apply** (*rule exI*) **using** *B1 B2 B3 B4 B5 B6* **by** *blast*  
**next**  
**assume** ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ( $\llbracket S \rrbracket$ )) ∈ *Rel*)  
 ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}  
 ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* → ( $\llbracket S \rrbracket$ , *T*) ∈ *TRel*)  
 ∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*  
 ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}

**from this obtain**  $Rel$  **where**  $C1: (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
**and**  $C2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3: (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
**and**  $C4: \text{weak-reduction-bisimulation } Rel\ (STCal\ Source\ Target)$  **and**  $C5: \text{preorder } Rel$   
**and**  $C6: \text{rel-weakly-respects-barb-set } Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**by** *auto*  
**from**  $C1\ C2\ C3\ C4\ C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge \text{preorder } Rel$   
 $\wedge \text{weak-reduction-bisimulation } Rel\ (STCal\ Source\ Target)$   
**by** *blast*  
**hence** *operational-corresponding*  $TRel \wedge \text{preorder } TRel \wedge \text{weak-reduction-bisimulation } TRel\ Target$   
**using** *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**moreover have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-weakly-respects-barb-set } Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**apply** (*rule exI*) **using**  $C1\ C6$  **by** *blast*  
**hence** *enc-weakly-respects-barb-set*  $\{success\}$   
**using** *success-sensitive-iff-source-target-rel-weakly-respects-success*  
**by** *simp*  
**moreover have** *rel-weakly-respects-barb-set*  $TRel\ TWB\ \{success\}$   
**proof** *auto*  
**fix**  $TP\ TQ\ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**fix**  $TP\ TQ\ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TQ \mapsto (Calculus\ TWB)^* TQ'$  **and**  $TQ' \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by** *blast*  
**thus**  $TP \downarrow \langle TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**qed**  
**ultimately show** *operational-corresponding*  $TRel \wedge \text{preorder } TRel$   
 $\wedge \text{weak-reduction-bisimulation } TRel\ Target$   
 $\wedge \text{enc-weakly-respects-barb-set } \{success\} \wedge \text{rel-weakly-respects-barb-set } TRel\ TWB\ \{success\}$   
**by** *fast*  
**qed**

**lemma** (**in** *encoding-wrt-barbs*) *OC-SS-RB-wrt-preorder-iff-source-target-rel*:  
**fixes**  $success :: 'barbs$   
**and**  $TRel :: ('procT \times 'procT)\ set$

**shows** (*operational-corresponding*  $TRel \wedge$  *preorder*  $TRel \wedge$  *weak-reduction-bisimulation*  $TRel$  *Target*  
 $\wedge$  *enc-weakly-respects-barbs*  $\wedge$  *rel-weakly-respects-barbs*  $TRel$  *TWB*  
 $\wedge$  *enc-weakly-respects-barb-set*  $\{success\}$   
 $\wedge$  *rel-weakly-respects-barb-set*  $TRel$  *TWB*  $\{success\}$ )  
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB* *TWB*)  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*)  $\{success\}$ )

**proof** (*rule iffI*, (*erule conjE*) $+$ )

**assume** *A1: rel-weakly-preserves-barbs*  $TRel$  *TWB* **and** *A2: rel-weakly-reflects-barbs*  $TRel$  *TWB*  
**and** *A3: enc-weakly-preserves-barbs* **and** *A4: enc-weakly-reflects-barbs*  
**and** *A5: preorder*  $TRel$

**from** *A5* **have** *A6:  $TRel^+ = TRel$*   
**using** *trancl-id*[*of*  $TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *blast*

**from** *A5* **have** *A7:  $TRel^* = TRel$*   
**using** *reflcl-trancl*[*of*  $TRel$ ] *trancl-id*[*of*  $TRel$ ]  
**unfolding** *preorder-on-def* *refl-on-def*  
**by** *auto*

**define**  $Rel$  **where**  $Rel = indRelRTPO\ TRel$

**hence** *B1:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$*   
**by** (*simp* *add: indRelRTPO.encR*)

**from** *Rel-def* *A6* **have** *B2:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$*   
**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]  
**by** (*auto* *simp* *add: indRelRTPO.target*)

**from** *Rel-def* *A7* **have** *B3:  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$*   
**using** *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]  
**by** *simp*

**assume** *operational-complete*  $TRel$  **and** *operational-sound*  $TRel$   
**and** *weak-reduction-simulation*  $TRel$  *Target*  
**and**  $\forall P\ Q\ Q'. (P, Q) \in TRel \wedge Q \longmapsto Target^* Q' \longrightarrow (\exists P'. P \longmapsto Target^* P' \wedge (P', Q') \in TRel)$

**with** *Rel-def* *A6* *A7* **have** *B4: weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)  
**using** *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*

**from** *Rel-def* *A5* **have** *B5: preorder*  $Rel$   
**using** *indRelRTPO-is-preorder*[**where**  $TRel = TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *blast*

**from** *Rel-def* *A1* *A2* *A3* *A4* **have** *B6: rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB* *TWB*)  
**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs*[**where**  $TRel = TRel$ ]  
**by** *blast*

**hence** *B7: rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*)  $\{success\}$   
**by** *blast*

**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB* *TWB*)  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*)  $\{success\}$

**apply** (*rule exI*) **using** *B1* *B2* *B3* *B4* *B5* *B6* *B7* **by** *blast*

**next**

**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *weak-reduction-bisimulation*  $Rel$  (*STCal* *Source* *Target*)  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB* *TWB*)  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*)  $\{success\}$

**from this obtain**  $Rel$  **where**  $C1: (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
**and**  $C2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3: (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
**and**  $C4: \text{weak-reduction-bisimulation } Rel\ (STCal\ Source\ Target)$  **and**  $C5: \text{preorder } Rel$   
**and**  $C6: \text{rel-weakly-respects-barbs } Rel\ (STCalWB\ SWB\ TWB)$   
**by** *auto*  
**from**  $C1\ C2\ C3\ C4\ C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge \text{preorder } Rel$   
 $\wedge \text{weak-reduction-bisimulation } Rel\ (STCal\ Source\ Target)$   
**by** *blast*  
**hence** *operational-corresponding*  $TRel \wedge \text{preorder } TRel \wedge \text{weak-reduction-bisimulation } TRel\ Target$   
**using** *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**moreover have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge \text{rel-weakly-respects-barbs } Rel\ (STCalWB\ SWB\ TWB)$   
**apply** (*rule exI*) **using**  $C1\ C6$  **by** *blast*  
**hence** *enc-weakly-respects-barbs*  
**using** *enc-weakly-respects-barbs-iff-source-target-rel*  
**by** *simp*  
**moreover hence** *enc-weakly-respects-barb-set*  $\{success\}$   
**by** *simp*  
**moreover have** *rel-weakly-respects-barbs*  $TRel\ TWB$   
**proof** *auto*  
**fix**  $TP\ TQ\ x\ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \downarrow \langle TWB \rangle x$   
**hence** *TargetTerm*  $TP \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have** *TargetTerm*  $TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using**  $C6$   
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**fix**  $TP\ TQ\ x\ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TQ \mapsto (Calculus\ TWB)^* TQ'$  **and**  $TQ' \downarrow \langle TWB \rangle x$   
**hence** *TargetTerm*  $TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have** *TargetTerm*  $TP \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using**  $C6$   
**by** *blast*  
**thus**  $TP \downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**qed**  
**moreover hence** *rel-weakly-respects-barb-set*  $TRel\ TWB\ \{success\}$   
**by** *blast*  
**ultimately show** *operational-corresponding*  $TRel \wedge \text{preorder } TRel$   
 $\wedge \text{weak-reduction-bisimulation } TRel\ Target$   
 $\wedge \text{enc-weakly-respects-barbs} \wedge \text{rel-weakly-respects-barbs } TRel\ TWB$   
 $\wedge \text{enc-weakly-respects-barb-set } \{success\} \wedge \text{rel-weakly-respects-barb-set } TRel\ TWB\ \{success\}$   
**by** *fast*

qed

An encoding is success sensitive and weakly operational corresponding w.r.t. a correspondence simulation  $TRel$  that respects success iff there exists a correspondence simulation that includes  $TRel$  and respects success. The same holds if we consider not only success sensitiveness but barb sensitiveness in general.

**lemma** (in *encoding-wrt-barbs*) *WOC-SS-wrt-preorder-iff-source-target-rel*:

**fixes** *success* :: 'barbs

**and** *TRel* :: ('procT × 'procT) set

**shows** (*weakly-operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $TRel$  *Target*  
 $\wedge$  *enc-weakly-respects-barb-set* {*success*}  
 $\wedge$  *rel-weakly-respects-barb-set*  $TRel$  *TWB* {*success*})  
 $=$  ( $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge$   $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge$  ( $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$ )  
 $\wedge$  *weak-reduction-correspondence-simulation*  $Rel$  (*STCal* *Source* *Target*)  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*) {*success*})

**proof** (*rule iffI*, (*erule conjE*) $+$ )

**assume** *A1*: *rel-weakly-preserves-barb-set*  $TRel$  *TWB* {*success*}

**and** *A2*: *rel-weakly-reflects-barb-set*  $TRel$  *TWB* {*success*}

**and** *A3*: *enc-weakly-preserves-barb-set* {*success*}

**and** *A4*: *enc-weakly-reflects-barb-set* {*success*}

**and** *A5*: *preorder*  $TRel$

**from** *A5* **have** *A6*:  $TRel^+ = TRel$

**using** *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def*

**by** *blast*

**from** *A5* *A6* **have** *A7*:  $TRel^* = TRel$

**using** *reflcl-trancl*[of  $TRel$ ] *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def* *refl-on-def*

**by** *auto*

**define** *Rel* **where**  $Rel = indRelRTPO\ TRel$

**hence** *B1*:  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

**by** (*simp* *add*: *indRelRTPO.encR*)

**from** *Rel-def* *A6* **have** *B2*:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]

**by** (*auto* *simp* *add*: *indRelRTPO.target*)

**from** *Rel-def* *A7* **have** *B3*:  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$

**using** *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ]

*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]

**by** *simp*

**assume** *operational-complete*  $TRel$  **and** *weakly-operational-sound*  $TRel$

**and** *weak-reduction-simulation*  $TRel$  *Target*

**and**  $\forall P\ Q\ Q'. (P, Q) \in TRel \wedge Q \longmapsto Target^*\ Q'$

$\longrightarrow (\exists P''\ Q''. P \longmapsto Target^*\ P'' \wedge Q' \longmapsto Target^*\ Q'' \wedge (P'', Q'') \in TRel)$

**with** *Rel-def* *A6* *A7* **have** *B4*: *weak-reduction-correspondence-simulation*  $Rel$  (*STCal* *Source* *Target*)

**using** *WOC-iff-indRelRTPO-is-reduction-correspondence-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**from** *Rel-def* *A5* **have** *B5*: *preorder*  $Rel$

**using** *indRelRTPO-is-preorder*[**where**  $TRel = TRel$ ]

**unfolding** *preorder-on-def*

**by** *blast*

**from** *Rel-def* *A1* *A2* *A3* *A4* **have** *B6*: *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB* *TWB*) {*success*}

**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*[**where**  $TRel = TRel$ ]

**and** *success=success*]

**by** *blast*

**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$

$\wedge$   $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

$\wedge$  ( $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$ )

$\wedge$  weak-reduction-correspondence-simulation  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  preorder  $Rel$   
 $\wedge$  rel-weakly-respects-barb-set  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**apply** (rule  $exI$ ) **using**  $B1$   $B2$   $B3$   $B4$   $B5$   $B6$  **by**  $blast$   
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  weak-reduction-correspondence-simulation  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  preorder  $Rel$   
 $\wedge$  rel-weakly-respects-barb-set  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**from**  $this$  **obtain**  $Rel$  **where**  $C1: (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
**and**  $C2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3: (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
**and**  $C4: weak-reduction-correspondence-simulation\ Rel$  ( $STCal$   $Source$   $Target$ )  
**and**  $C5: preorder\ Rel$  **and**  $C6: rel-weakly-respects-barb-set\ Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**by**  $auto$   
**from**  $C1$   $C2$   $C3$   $C4$   $C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge$  preorder  $Rel$   
 $\wedge$  weak-reduction-correspondence-simulation  $Rel$  ( $STCal$   $Source$   $Target$ )  
**by**  $blast$   
**hence** weakly-operational-corresponding  $TRel \wedge$  preorder  $TRel$   
 $\wedge$  weak-reduction-correspondence-simulation  $TRel$   $Target$   
**using**  $WOC-wrt-preorder-iff-reduction-correspondence-simulation$  [**where**  $TRel=TRel$ ]  
**by**  $simp$   
**moreover** **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge$  rel-weakly-respects-barb-set  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**apply** (rule  $exI$ ) **using**  $C1$   $C6$  **by**  $blast$   
**hence** enc-weakly-respects-barb-set  $\{success\}$   
**using**  $success-sensitive-iff-source-target-rel-weakly-respects-success$   
**by**  $simp$   
**moreover** **have** rel-weakly-respects-barb-set  $TRel$   $TWB$   $\{success\}$   
**proof**  $auto$   
**fix**  $TP$   $TQ$   $TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by**  $simp$   
**moreover** **assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $STCalWB-reachesBarbST$   
**by**  $blast$   
**ultimately** **have**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by**  $blast$   
**thus**  $TQ \downarrow \langle TWB \rangle success$   
**using**  $STCalWB-reachesBarbST$   
**by**  $blast$   
**next**  
**fix**  $TP$   $TQ$   $TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by**  $simp$   
**moreover** **assume**  $TQ \mapsto (Calculus\ TWB)^* TQ'$  **and**  $TQ' \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $STCalWB-reachesBarbST$   
**by**  $blast$   
**ultimately** **have**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using**  $C6$   
**by**  $blast$   
**thus**  $TP \downarrow \langle TWB \rangle success$   
**using**  $STCalWB-reachesBarbST$   
**by**  $blast$

qed  
**ultimately show** *weakly-operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $TRel$  *Target*  
 $\wedge$  *enc-weakly-respects-barb-set*  $\{success\} \wedge$  *rel-weakly-respects-barb-set*  $TRel$   $TWB$   $\{success\}$   
**by** *fast*  
qed

**lemma** (in *encoding-wrt-barbs*) *WOC-SS-RB-wrt-preorder-iff-source-target-rel*:

**fixes** *success* :: 'barbs

**and**  $TRel$  :: ('procT  $\times$  'procT) set

**shows** (*weakly-operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$

$\wedge$  *weak-reduction-correspondence-simulation*  $TRel$  *Target*

$\wedge$  *enc-weakly-respects-barbs*  $\wedge$  *enc-weakly-respects-barb-set*  $\{success\}$

$\wedge$  *rel-weakly-respects-barbs*  $TRel$   $TWB \wedge$  *rel-weakly-respects-barb-set*  $TRel$   $TWB$   $\{success\}$ )

=  $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$

$\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$

$\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$

$\wedge$  *weak-reduction-correspondence-simulation*  $Rel$  (*STCal* *Source* *Target*)  $\wedge$  *preorder*  $Rel$

$\wedge$  *rel-weakly-respects-barbs*  $Rel$  (*STCalWB* *SWB*  $TWB$ )

$\wedge$  *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB*  $TWB$ )  $\{success\}$ )

**proof** (*rule iffI*, (*erule conjE*) $+$ )

**assume**  $A1$ : *rel-weakly-preserves-barb-set*  $TRel$   $TWB$   $\{success\}$

**and**  $A2$ : *rel-weakly-reflects-barb-set*  $TRel$   $TWB$   $\{success\}$

**and**  $A3$ : *enc-weakly-preserves-barb-set*  $\{success\}$

**and**  $A4$ : *enc-weakly-reflects-barb-set*  $\{success\}$

**and**  $A5$ : *preorder*  $TRel$

**and**  $A1'$ : *rel-weakly-preserves-barbs*  $TRel$   $TWB$  **and**  $A2'$ : *rel-weakly-reflects-barbs*  $TRel$   $TWB$

**and**  $A3'$ : *enc-weakly-preserves-barbs* **and**  $A4'$ : *enc-weakly-reflects-barbs*

**from**  $A5$  **have**  $A6$ :  $TRel^+ = TRel$

**using** *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def*

**by** *blast*

**from**  $A5$   $A6$  **have**  $A7$ :  $TRel^* = TRel$

**using** *reflcl-trancl*[of  $TRel$ ] *trancl-id*[of  $TRel$ ]

**unfolding** *preorder-on-def* *refl-on-def*

**by** *auto*

**define**  $Rel$  **where**  $Rel = indRelRTPO$   $TRel$

**hence**  $B1$ :  $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel$

**by** (*simp add*: *indRelRTPO.encR*)

**from**  $Rel$ -*def*  $A6$  **have**  $B2$ :  $TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$

**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]

**by** (*auto simp add*: *indRelRTPO.target*)

**from**  $Rel$ -*def*  $A7$  **have**  $B3$ :  $\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$

**using** *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ]

*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]

**by** *simp*

**assume** *operational-complete*  $TRel$  **and** *weakly-operational-sound*  $TRel$

**and** *weak-reduction-simulation*  $TRel$  *Target*

**and**  $\forall P Q Q'. (P, Q) \in TRel \wedge Q \longmapsto Target^* Q'$

$\longrightarrow (\exists P'' Q''. P \longmapsto Target^* P'' \wedge Q' \longmapsto Target^* Q'' \wedge (P'', Q'') \in TRel)$

**with**  $Rel$ -*def*  $A6$   $A7$  **have**  $B4$ : *weak-reduction-correspondence-simulation*  $Rel$  (*STCal* *Source* *Target*)

**using** *WOC-iff-indRelRTPO-is-reduction-correspondence-simulation*[**where**  $TRel = TRel$ ]

**by** *simp*

**from**  $Rel$ -*def*  $A5$  **have**  $B5$ : *preorder*  $Rel$

**using** *indRelRTPO-is-preorder*[**where**  $TRel = TRel$ ]

**unfolding** *preorder-on-def*

**by** *blast*

**from**  $Rel$ -*def*  $A1$   $A2$   $A3$   $A4$  **have**  $B6$ : *rel-weakly-respects-barb-set*  $Rel$  (*STCalWB* *SWB*  $TWB$ )  $\{success\}$

**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*[**where**  $TRel = TRel$

**and** *success=success*]

**by** *blast*



**from** *Rel-def*  $A1' A2' A3' A4'$  **have**  $B7$ : *rel-weakly-respects-barbs*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  
**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs*[**where**  $TRel = TRel$ ]  
**by** *blast*  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**apply** (*rule exI*) **using**  $B1\ B2\ B3\ B4\ B5\ B6\ B7$  **by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  
 $\wedge$  *rel-weakly-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**from** *this* **obtain**  $Rel$  **where**  $C1$ :  $(\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
**and**  $C2$ :  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3$ :  $(\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
**and**  $C4$ : *weak-reduction-correspondence-simulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**and**  $C5$ : *preorder*  $Rel$  **and**  $C7$ : *rel-weakly-respects-barbs*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  
**by** *auto*  
**from**  $C1\ C2\ C3\ C4\ C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge$  *preorder*  $Rel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**by** *blast*  
**hence** *weakly-operational-corresponding*  $TRel \wedge$  *preorder*  $TRel$   
 $\wedge$  *weak-reduction-correspondence-simulation*  $TRel$   $Target$   
**using** *WOC-wrt-preorder-iff-reduction-correspondence-simulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**moreover** **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge$  *rel-weakly-respects-barbs*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  
**apply** (*rule exI*) **using**  $C1\ C7$  **by** *blast*  
**hence**  $D1$ : *enc-weakly-respects-barbs*  
**using** *enc-weakly-respects-barbs-iff-source-target-rel*  
**by** *simp*  
**moreover** **from**  $D1$  **have** *enc-weakly-respects-barb-set*  $\{success\}$   
**by** *simp*  
**moreover** **have**  $D2$ : *rel-weakly-respects-barbs*  $TRel\ TWB$   
**proof** *auto*  
**fix**  $TP\ TQ\ x\ TP'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \Downarrow \langle TWB \rangle x$   
**hence**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately** **have**  $TargetTerm\ TQ \Downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
**using**  $C7$   
**by** *blast*  
**thus**  $TQ \Downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**fix**  $TP\ TQ\ x\ TQ'$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*

**moreover assume**  $TQ \mapsto (\text{Calculus } TWB)^* TQ'$  **and**  $TQ' \Downarrow \langle TWB \rangle x$   
**hence**  $\text{TargetTerm } TQ \Downarrow \langle \text{STCalWB } SWB \text{ TWB} \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $\text{TargetTerm } TP \Downarrow \langle \text{STCalWB } SWB \text{ TWB} \rangle x$   
**using** *C7*  
**by** *blast*  
**thus**  $TP \Downarrow \langle TWB \rangle x$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**qed**  
**moreover from** *D2* **have**  $\text{rel-weakly-respects-barb-set } TRel \text{ TWB } \{success\}$   
**by** *blast*  
**ultimately show**  $\text{weakly-operational-corresponding } TRel \wedge \text{preorder } TRel$   
 $\wedge \text{weak-reduction-correspondence-simulation } TRel \text{ Target}$   
 $\wedge \text{enc-weakly-respects-barbs} \wedge \text{enc-weakly-respects-barb-set } \{success\}$   
 $\wedge \text{rel-weakly-respects-barbs } TRel \text{ TWB} \wedge \text{rel-weakly-respects-barb-set } TRel \text{ TWB } \{success\}$   
**by** *fast*  
**qed**

An encoding is strongly success sensitive and strongly operational corresponding w.r.t. a strong bisimulation  $TRel$  that strongly respects success iff there exists a strong bisimulation that includes  $TRel$  and strongly respects success. The same holds if we consider not only strong success sensitiveness but strong barb sensitiveness in general.

**lemma** (in *encoding-wrt-barbs*) *SOC-SS-wrt-preorder-iff-source-target-rel*:

**fixes**  $success :: 'barbs$

**and**  $TRel :: ('procT \times 'procT) \text{ set}$

**shows** ( $\text{strongly-operational-corresponding } TRel \wedge \text{preorder } TRel$

$\wedge \text{strong-reduction-bisimulation } TRel \text{ Target}$

$\wedge \text{enc-respects-barb-set } \{success\} \wedge \text{rel-respects-barb-set } TRel \text{ TWB } \{success\}$ )

$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$

$\wedge \text{strong-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target}) \wedge \text{preorder } \text{Rel}$

$\wedge \text{rel-respects-barb-set } \text{Rel } (\text{STCalWB } SWB \text{ TWB}) \{success\}$ )

**proof** (*rule iffI*, (*erule conjE*)+)

**assume**  $A1: \text{rel-preserves-barb-set } TRel \text{ TWB } \{success\}$

**and**  $A2: \text{rel-reflects-barb-set } TRel \text{ TWB } \{success\}$

**and**  $A3: \text{enc-preserves-barb-set } \{success\}$  **and**  $A4: \text{enc-reflects-barb-set } \{success\}$

**and**  $A5: \text{preorder } TRel$

**from**  $A5$  **have**  $A6: TRel^+ = TRel$

**using** *trancl-id[of TRel]*

**unfolding** *preorder-on-def*

**by** *blast*

**from**  $A5$   $A6$  **have**  $A7: TRel^* = TRel$

**using** *reflcl-trancl[of TRel]* *trancl-id[of TRel]*

**unfolding** *preorder-on-def* *refl-on-def*

**by** *auto*

**define**  $\text{Rel}$  **where**  $\text{Rel} = \text{indRelRTPO } TRel$

**hence**  $B1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

**by** (*simp add: indRelRTPO.encR*)

**from** *Rel-def*  $A6$  **have**  $B2: TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

**using** *indRelRTPO-to-TRel(4)[where TRel=TRel]*

**by** (*auto simp add: indRelRTPO.target*)

**from** *Rel-def*  $A7$  **have**  $B3: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in TRel$

**using** *indRelRTPO-to-TRel(2)[where TRel=TRel]*

*trans-closure-of-TRel-refl-cond[where TRel=TRel]*

**by** *simp*

**assume**  $\text{strongly-operational-complete } TRel$  **and**  $\text{strongly-operational-sound } TRel$

**and**  $\text{strong-reduction-simulation } TRel \text{ Target}$

**and**  $\forall P Q Q'. (P, Q) \in TRel \wedge Q \mapsto Target Q' \longrightarrow (\exists P'. P \mapsto Target P' \wedge (P', Q') \in TRel)$   
**with** *Rel-def A6 A7* **have**  $B_4$ : *strong-reduction-bisimulation Rel (STCal Source Target)*  
**using** *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**from** *Rel-def A5* **have**  $B_5$ : *preorder Rel*  
**using** *indRelRTPO-is-preorder*[**where**  $TRel = TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *blast*  
**from** *Rel-def A1 A2 A3 A4* **have**  $B_6$ : *rel-respects-barb-set Rel (STCalWB SWB TWB) {success}*  
**using** *enc-and-TRel-impl-indRelRTPO-respects-success*[**where**  $TRel = TRel$  **and**  $success = success$ ]  
**by** *blast*  
**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge strong-reduction-bisimulation Rel (STCal Source Target) \wedge preorder Rel$   
 $\wedge rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}$   
**apply** (*rule exI*) **using**  $B_1 B_2 B_3 B_4 B_5 B_6$  **by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge strong-reduction-bisimulation Rel (STCal Source Target) \wedge preorder Rel$   
 $\wedge rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}$   
**from** *this* **obtain** *Rel* **where**  $C_1$ :  $(\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $C_2$ :  $TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$   
 $C_3$ :  $(\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $C_4$ : *strong-reduction-bisimulation Rel (STCal Source Target)* **and**  $C_5$ : *preorder Rel*  
 $C_6$ : *rel-respects-barb-set Rel (STCalWB SWB TWB) {success}*  
**by** *auto*  
**from**  $C_1 C_2 C_3 C_4 C_5$  **have**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge (TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\})$   
 $\wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge preorder Rel$   
 $\wedge strong-reduction-bisimulation Rel (STCal Source Target)$   
**by** *blast*  
**hence** *strongly-operational-corresponding*  $TRel \wedge preorder TRel$   
 $\wedge strong-reduction-bisimulation TRel Target$   
**using** *SOC-wrt-preorder-iff-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**moreover** **have**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}$   
**apply** (*rule exI*) **using**  $C_1 C_6$  **by** *blast*  
**hence** *enc-respects-barb-set {success}*  
**using** *success-sensitive-iff-source-target-rel-respects-success*  
**by** *simp*  
**moreover** **have** *rel-respects-barb-set*  $TRel TWB \{success\}$   
**proof** *auto*  
**fix**  $TP TQ$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C_2$  **have**  $(TargetTerm TP, TargetTerm TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TP \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm TP \downarrow \langle STCalWB SWB TWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**ultimately** **have**  $TargetTerm TQ \downarrow \langle STCalWB SWB TWB \rangle success$   
**using**  $C_6$   
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle success$   
**using** *STCalWB-hasBarbST*  
**by** *blast*  
**next**

**fix**  $TP\ TQ$   
**assume**  $(TP, TQ) \in TRel$   
**with**  $C2$  **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
    **by** *simp*  
**moreover assume**  $TQ \downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
    **using** *STCalWB-hasBarbST*  
    **by** *blast*  
**ultimately have**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
    **using** *C6*  
    **by** *blast*  
**thus**  $TP \downarrow \langle TWB \rangle success$   
    **using** *STCalWB-hasBarbST*  
    **by** *blast*  
**qed**  
**ultimately show** *strongly-operational-corresponding*  $TRel \wedge preorder\ TRel$   
 $\wedge$  *strong-reduction-bisimulation*  $TRel\ Target$   
 $\wedge$  *enc-respects-barb-set*  $\{success\} \wedge rel-respects-barb-set\ TRel\ TWB\ \{success\}$   
    **by** *fast*  
**qed**

**lemma** (in *encoding-wrt-barbs*) *SOC-SS-RB-wrt-preorder-iff-source-target-rel*:

**fixes** *success* :: 'barbs  
**and**  $TRel$  :: ('procT  $\times$  'procT) set  
**shows** (*strongly-operational-corresponding*  $TRel \wedge preorder\ TRel$   
 $\wedge$  *strong-reduction-bisimulation*  $TRel\ Target$   
 $\wedge$  *enc-respects-barbs*  $\wedge rel-respects-barbs\ TRel\ TWB$   
 $\wedge$  *enc-respects-barb-set*  $\{success\} \wedge rel-respects-barb-set\ TRel\ TWB\ \{success\}$ )  
=  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *strong-reduction-bisimulation*  $Rel\ (STCal\ Source\ Target) \wedge preorder\ Rel$   
 $\wedge rel-respects-barbs\ Rel\ (STCalWB\ SWB\ TWB)$   
 $\wedge rel-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\})$

**proof** (*rule iffI, (erule conjE)+*)

**assume**  $A1$ : *rel-preserves-barbs*  $TRel\ TWB$  **and**  $A2$ : *rel-reflects-barbs*  $TRel\ TWB$   
**and**  $A3$ : *enc-preserves-barbs* **and**  $A4$ : *enc-reflects-barbs*  
**and**  $A5$ : *preorder*  $TRel$   
**from**  $A5$  **have**  $A6$ :  $TRel^+ = TRel$   
    **using** *trancl-id[of TRel]*  
    **unfolding** *preorder-on-def*  
    **by** *blast*  
**from**  $A5$  **have**  $A7$ :  $TRel^* = TRel$   
    **using** *reflcl-trancl[of TRel] trancl-id[of TRel]*  
    **unfolding** *preorder-on-def refl-on-def*  
    **by** *auto*

**define**  $Rel$  **where**  $Rel = indRelRTPO\ TRel$

**hence**  $B1$ :  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$

**by** (*simp add: indRelRTPO.encR*)

**from** *Rel-def A6* **have**  $B2$ :  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

**using** *indRelRTPO-to-TRel(4)[where TRel=TRel]*

**by** (*auto simp add: indRelRTPO.target*)

**from** *Rel-def A7* **have**  $B3$ :  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$

**using** *indRelRTPO-to-TRel(2)[where TRel=TRel]*

*trans-closure-of-TRel-refl-cond[where TRel=TRel]*

**by** *simp*

**assume** *strongly-operational-complete*  $TRel$  **and** *strongly-operational-sound*  $TRel$

**and** *strong-reduction-simulation*  $TRel\ Target$

**and**  $\forall P\ Q\ Q'. (P, Q) \in TRel \wedge Q \longmapsto Target\ Q' \longrightarrow (\exists P'. P \longmapsto Target\ P' \wedge (P', Q') \in TRel)$

**with** *Rel-def A6 A7* **have**  $B4$ : *strong-reduction-bisimulation*  $Rel\ (STCal\ Source\ Target)$

**using** *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]*

**by** *simp*  
**from** *Rel-def A5* **have** *B5: preorder Rel*  
    **using** *indRelRTPO-is-preorder*[**where** *TRel=TRel*]  
    **unfolding** *preorder-on-def*  
**by** *blast*  
**from** *Rel-def A1 A2 A3 A4* **have** *B6: rel-respects-barbs Rel (STCalWB SWB TWB)*  
    **using** *enc-and-TRel-impl-indRelRTPO-respects-barbs*[**where** *TRel=TRel*]  
**by** *blast*  
**hence** *B7: rel-respects-barb-set Rel (STCalWB SWB TWB) {success}*  
**by** *blast*  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
     $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
     $\wedge strong\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target) \wedge preorder\ Rel$   
     $\wedge rel\text{-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
     $\wedge rel\text{-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**apply** (*rule exI*) **using** *B1 B2 B3 B4 B5 B6* **by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
     $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
     $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
     $\wedge strong\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target) \wedge preorder\ Rel$   
     $\wedge rel\text{-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
     $\wedge rel\text{-respects-barb-set}\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**from** *this* **obtain** *Rel* **where** *C1: ( $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$ )*  
**and** *C2:  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$*   
**and** *C3: ( $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$ )*  
**and** *C4: strong-reduction-bisimulation Rel (STCal Source Target)* **and** *C5: preorder Rel*  
**and** *C6: rel-respects-barbs Rel (STCalWB SWB TWB)*  
**by** *auto*  
**from** *C1 C2 C3 C4 C5* **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
     $\wedge (TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\})$   
     $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge preorder\ Rel$   
     $\wedge strong\text{-reduction-bisimulation}\ Rel\ (STCal\ Source\ Target)$   
**by** *blast*  
**hence** *strongly-operational-corresponding TRel  $\wedge$  preorder TRel*  
     $\wedge strong\text{-reduction-bisimulation}\ TRel\ Target$   
    **using** *SOC-wrt-preorder-iff-strong-reduction-bisimulation*[**where** *TRel=TRel*]  
**by** *simp*  
**moreover** **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
     $\wedge rel\text{-respects-barbs}\ Rel\ (STCalWB\ SWB\ TWB)$   
    **apply** (*rule exI*) **using** *C1 C6* **by** *blast*  
**hence** *enc-respects-barbs*  
    **using** *enc-respects-barbs-iff-source-target-rel*  
**by** *simp*  
**moreover** **hence** *enc-respects-barb-set {success}*  
**by** *simp*  
**moreover** **have** *rel-respects-barbs TRel TWB*  
**proof** *auto*  
**fix** *TP TQ x*  
**assume**  $(TP, TQ) \in TRel$   
**with** *C2* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover** **assume**  $TP \downarrow \langle TWB \rangle x$   
**hence**  $TargetTerm\ TP \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
    **using** *STCalWB-hasBarbST*  
**by** *blast*  
**ultimately** **have**  $TargetTerm\ TQ \downarrow \langle STCalWB\ SWB\ TWB \rangle x$   
    **using** *C6*  
**by** *blast*  
**thus**  $TQ \downarrow \langle TWB \rangle x$

```

    using STCalWB-hasBarbST
  by blast
next
fix TP TQ x
assume (TP, TQ) ∈ TRel
with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
  by simp
moreover assume TQ ↓ <TWB> x
hence TargetTerm TQ ↓ <STCalWB SWB TWB> x
  using STCalWB-hasBarbST
  by blast
ultimately have TargetTerm TP ↓ <STCalWB SWB TWB> x
  using C6
  by blast
thus TP ↓ <TWB> x
  using STCalWB-hasBarbST
  by blast
qed
moreover hence rel-respects-barb-set TRel TWB {success}
  by blast
ultimately show strongly-operational-corresponding TRel ∧ preorder TRel
  ∧ strong-reduction-bisimulation TRel Target
  ∧ enc-respects-barbs ∧ rel-respects-barbs TRel TWB
  ∧ enc-respects-barb-set {success} ∧ rel-respects-barb-set TRel TWB {success}
  by fast
qed

```

Next we also add divergence reflection to operational correspondence and success sensitiveness.

```

lemma (in encoding) enc-and-TRelimpl-indRelRTPO-reflect-divergence:
  fixes TRel :: ('procT × 'procT) set
  assumes encRD: enc-reflects-divergence
    and trelRD: rel-reflects-divergence TRel Target
  shows rel-reflects-divergence (indRelRTPO TRel) (STCal Source Target)
proof auto
  fix P Q
  assume P ≲ [·] RT <TRel> Q and Q ⟶ (STCal Source Target) ω
  thus P ⟶ (STCal Source Target) ω
proof induct
  case (encR S)
  assume TargetTerm ([S]) ⟶ (STCal Source Target) ω
  hence [S] ⟶ (Target) ω
    by (simp add: STCal-divergent(2))
  with encRD have S ⟶ (Source) ω
    by simp
  thus SourceTerm S ⟶ (STCal Source Target) ω
    by (simp add: STCal-divergent(1))
next
  case (source S)
  assume SourceTerm S ⟶ (STCal Source Target) ω
  thus SourceTerm S ⟶ (STCal Source Target) ω
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  moreover assume TargetTerm T2 ⟶ (STCal Source Target) ω
  hence T2 ⟶ (Target) ω
    by (simp add: STCal-divergent(2))
  ultimately have T1 ⟶ (Target) ω
    using trelRD
    by blast
  thus TargetTerm T1 ⟶ (STCal Source Target) ω

```

```

  by (simp add: STCal-divergent(2))
next
case (trans P Q R)
assume R  $\mapsto$ (STCal Source Target) $\omega$ 
  and R  $\mapsto$ (STCal Source Target) $\omega \implies Q \mapsto$ (STCal Source Target) $\omega$ 
  and Q  $\mapsto$ (STCal Source Target) $\omega \implies P \mapsto$ (STCal Source Target) $\omega$ 
thus P  $\mapsto$ (STCal Source Target) $\omega$ 
  by simp
qed
qed

```

**lemma** (in *encoding-wrt-barbs*) *OC-SS-DR-iff-source-target-rel*:

```

fixes success :: 'barbs
and TRel :: ('procT  $\times$  'procT) set
shows (operational-corresponding (TRel*)
   $\wedge$  weak-reduction-bisimulation (TRel+) Target
   $\wedge$  enc-weakly-respects-barb-set {success}
   $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
   $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence TRel Target)
= ( $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)
   $\wedge$  ( $\forall T1 T2$ . (T1, T2)  $\in$  TRel  $\longrightarrow$  (TargetTerm T1, TargetTerm T2)  $\in$  Rel)
   $\wedge$  ( $\forall T1 T2$ . (TargetTerm T1, TargetTerm T2)  $\in$  Rel  $\longrightarrow$  (T1, T2)  $\in$  TRel+)
   $\wedge$  ( $\forall S T$ . (SourceTerm S, TargetTerm T)  $\in$  Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T)  $\in$  TRel*)
   $\wedge$  weak-reduction-bisimulation Rel (STCal Source Target)
   $\wedge$  rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
   $\wedge$  rel-reflects-divergence Rel (STCal Source Target))
proof (rule iffI, (erule conjE)+)
  assume A1: rel-weakly-preserves-barb-set TRel TWB {success}
  and A2: rel-weakly-reflects-barb-set TRel TWB {success}
  and A3: enc-weakly-preserves-barb-set {success}
  and A4: enc-weakly-reflects-barb-set {success}
  and A5: rel-reflects-divergence TRel Target and A6: enc-reflects-divergence
define Rel where Rel = indRelRTPO TRel
hence B1:  $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel
  by (simp add: indRelRTPO.encR)
from Rel-def have B2:  $\forall T1 T2$ . (T1, T2)  $\in$  TRel  $\longrightarrow$  (TargetTerm T1, TargetTerm T2)  $\in$  Rel
  by (simp add: indRelRTPO.target)
from Rel-def have B3:  $\forall T1 T2$ . (TargetTerm T1, TargetTerm T2)  $\in$  Rel  $\longrightarrow$  (T1, T2)  $\in$  TRel+
  by (simp add: indRelRTPO-to-TRel(4)[where TRel=TRel])
from Rel-def have B4:  $\forall S T$ . (SourceTerm S, TargetTerm T)  $\in$  Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T)  $\in$  TRel*
  using indRelRTPO-to-TRel(2)[where TRel=TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  by simp
assume operational-complete (TRel*)
  and operational-sound (TRel*)
  and weak-reduction-simulation (TRel+) Target
  and  $\forall P Q Q'$ . (P, Q)  $\in$  TRel+  $\wedge$  Q  $\mapsto$ Target* Q'
   $\longrightarrow$  ( $\exists P'$ . P  $\mapsto$ Target* P'  $\wedge$  (P', Q')  $\in$  TRel+)
with Rel-def have B5: weak-reduction-bisimulation Rel (STCal Source Target)
  using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel=TRel]
  by simp
from Rel-def A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel]
  and success=success]
  by blast
from Rel-def A5 A6 have B7: rel-reflects-divergence Rel (STCal Source Target)
  using enc-and-TRelimpl-indRelRTPO-reflect-divergence[where TRel=TRel]
  by blast
show  $\exists$  Rel. ( $\forall S$ . (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)
   $\wedge$  ( $\forall T1 T2$ . (T1, T2)  $\in$  TRel  $\longrightarrow$  (TargetTerm T1, TargetTerm T2)  $\in$  Rel)
   $\wedge$  ( $\forall T1 T2$ . (TargetTerm T1, TargetTerm T2)  $\in$  Rel  $\longrightarrow$  (T1, T2)  $\in$  TRel+)

```

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-bisimulation } \text{Rel} \text{ (STCal Source Target)}$   
 $\wedge \text{rel-weakly-respects-barb-set } \text{Rel} \text{ (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence } \text{Rel} \text{ (STCal Source Target)}$   
**apply** (rule exI) **using** B1 B2 B3 B4 B5 B6 B7 **by** blast  
**next**  
**assume**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-bisimulation } \text{Rel} \text{ (STCal Source Target)}$   
 $\wedge \text{rel-weakly-respects-barb-set } \text{Rel} \text{ (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence } \text{Rel} \text{ (STCal Source Target)}$   
**from** this **obtain** Rel **where** C1:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$   
**and** C2:  $\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$   
**and** C3:  $\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+$   
**and** C4:  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$   
**and** C5: weak-reduction-bisimulation Rel (STCal Source Target)  
**and** C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}  
**and** C7: rel-reflects-divergence Rel (STCal Source Target)  
**by** auto  
**from** C1 C2 C3 C4 C5 **have**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$   
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$   
 $\wedge \text{weak-reduction-bisimulation } \text{Rel} \text{ (STCal Source Target)}$   
**by** blast  
**hence** operational-corresponding (TRel\*)  
 $\wedge \text{weak-reduction-bisimulation } (\text{TRel}^+) \text{ Target}$   
**using** OC-iff-weak-reduction-bisimulation[**where** TRel=TRel]  
**by** auto  
**moreover** **have**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{rel-weakly-respects-barb-set } \text{Rel} \text{ (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence } \text{Rel} \text{ (STCal Source Target)}$   
**apply** (rule exI) **using** C1 C6 C7 **by** blast  
**hence** enc-weakly-respects-barb-set {success}  $\wedge$  enc-reflects-divergence  
**using** WSS-DR-iff-source-target-rel  
**by** auto  
**moreover** **have** rel-weakly-respects-barb-set TRel TWB {success}  
**proof** auto  
**fix** TP TQ TP'  
**assume**  $(TP, TQ) \in \text{TRel}$   
**with** C2 **have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$   
**by** simp  
**moreover** **assume**  $TP \mapsto (\text{Calculus } \text{TWB})^* TP'$  **and**  $TP' \Downarrow \langle \text{TWB} \rangle \text{success}$   
**hence**  $\text{TargetTerm } TP \Downarrow \langle \text{STCalWB } \text{SWB } \text{TWB} \rangle \text{success}$   
**using** STCalWB-reachesBarbST  
**by** blast  
**ultimately** **have**  $\text{TargetTerm } TQ \Downarrow \langle \text{STCalWB } \text{SWB } \text{TWB} \rangle \text{success}$   
**using** C6  
**by** blast  
**thus**  $TQ \Downarrow \langle \text{TWB} \rangle \text{success}$   
**using** STCalWB-reachesBarbST  
**by** blast  
**next**  
**fix** TP TQ TQ'  
**assume**  $(TP, TQ) \in \text{TRel}$   
**with** C2 **have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$   
**by** simp  
**moreover** **assume**  $TQ \mapsto (\text{Calculus } \text{TWB})^* TQ'$  **and**  $TQ' \Downarrow \langle \text{TWB} \rangle \text{success}$   
**hence**  $\text{TargetTerm } TQ \Downarrow \langle \text{STCalWB } \text{SWB } \text{TWB} \rangle \text{success}$



```

    using STCalWB-reachesBarbST
  by blast
ultimately have TargetTerm TP $\Downarrow$ <STCalWB SWB TWB>success
  using C6
  by blast
thus TP $\Downarrow$ <TWB>success
  using STCalWB-reachesBarbST
  by blast
qed
moreover from C2 C7 have rel-reflects-divergence TRel Target
  using STCal-divergent(2)
  by blast
ultimately show operational-corresponding (TRel*)
 $\wedge$  weak-reduction-bisimulation (TRel+) Target
 $\wedge$  enc-weakly-respects-barb-set {success}  $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
 $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence TRel Target
  by fast
qed

lemma (in encoding-wrt-barbs) WOC-SS-DR-wrt-preorder-iff-source-target-rel:
fixes success :: 'barbs
  and TRel :: ('procT  $\times$  'procT) set
shows (weakly-operational-corresponding TRel  $\wedge$  preorder TRel
 $\wedge$  weak-reduction-correspondence-simulation TRel Target
 $\wedge$  enc-weakly-respects-barb-set {success}
 $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
 $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence TRel Target)
= ( $\exists$  Rel. ( $\forall$  S. (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)
 $\wedge$  TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2)  $\in$  Rel}
 $\wedge$  ( $\forall$  S T. (SourceTerm S, TargetTerm T)  $\in$  Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T)  $\in$  TRel)
 $\wedge$  weak-reduction-correspondence-simulation Rel (STCal Source Target)  $\wedge$  preorder Rel
 $\wedge$  rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
 $\wedge$  rel-reflects-divergence Rel (STCal Source Target))
proof (rule iffI, (erule conjE)+)
  assume A1: rel-weakly-preserves-barb-set TRel TWB {success}
  and A2: rel-weakly-reflects-barb-set TRel TWB {success}
  and A3: enc-weakly-preserves-barb-set {success}
  and A4: enc-weakly-reflects-barb-set {success}
  and A5: rel-reflects-divergence TRel Target and A6: enc-reflects-divergence
  and A7: preorder TRel
  from A7 have A8: TRel+ = TRel
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
  from A7 have A9: TRel* = TRel
  using refl-trancl[of TRel] trancl-id[of TRel]
  unfolding preorder-on-def refl-on-def
  by auto
  define Rel where Rel = indRelRTPO TRel
  hence B1:  $\forall$  S. (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel
  by (simp add: indRelRTPO.encR)
  from Rel-def A8 have B2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2)  $\in$  Rel}
  using indRelRTPO-to-TRel(4)[where TRel=TRel]
  by (auto simp add: indRelRTPO.target)
  from Rel-def A9 have B3:  $\forall$  S T. (SourceTerm S, TargetTerm T)  $\in$  Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T)  $\in$  TRel
  using indRelRTPO-to-TRel(2)[where TRel=TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  by simp
  assume operational-complete TRel and weakly-operational-sound TRel and preorder TRel
  and weak-reduction-simulation TRel Target
  and  $\forall P Q Q'. (P, Q) \in TRel \wedge Q \longmapsto Target* Q'$ 

```

$\longrightarrow (\exists P'' Q''. P \longmapsto \text{Target} * P'' \wedge Q' \longmapsto \text{Target} * Q'' \wedge (P'', Q'') \in \text{TRel})$   
**with** *Rel-def A8 A9* **have** *B4: weak-reduction-correspondence-simulation Rel (STCal Source Target)*  
**using** *WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where TRel=TRel]*  
**by** *simp*  
**from** *Rel-def A7* **have** *B5: preorder Rel*  
**using** *indRelRTPO-is-preorder[where TRel=TRel]*  
**unfolding** *preorder-on-def*  
**by** *simp*  
**from** *Rel-def A1 A2 A3 A4* **have** *B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}*  
**using** *enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel]*  
**and** *success=success*  
**by** *blast*  
**from** *Rel-def A5 A6* **have** *B7: rel-reflects-divergence Rel (STCal Source Target)*  
**using** *enc-and-TRelimpl-indRelRTPO-reflect-divergence[where TRel=TRel]*  
**by** *blast*  
**show**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$   
 $\wedge \text{weak-reduction-correspondence-simulation Rel (STCal Source Target)} \wedge \text{preorder Rel}$   
 $\wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence Rel (STCal Source Target)}$   
**apply** (*rule exI*) **using** *B1 B2 B3 B4 B5 B6 B7* **by** *blast*  
**next**  
**assume**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$   
 $\wedge \text{weak-reduction-correspondence-simulation Rel (STCal Source Target)} \wedge \text{preorder Rel}$   
 $\wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence Rel (STCal Source Target)}$   
**from** *this* **obtain** *Rel* **where** *C1:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$*   
**and** *C2:  $\text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$*   
**and** *C3:  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}$*   
**and** *C4: weak-reduction-correspondence-simulation Rel (STCal Source Target)*  
**and** *C5: preorder Rel* **and** *C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}*  
**and** *C7: rel-reflects-divergence Rel (STCal Source Target)*  
**by** *auto*  
**from** *C1 C2 C3 C4 C5* **have**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$   
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}) \wedge \text{preorder Rel}$   
 $\wedge \text{weak-reduction-correspondence-simulation Rel (STCal Source Target)}$   
**by** *blast*  
**hence** *weakly-operational-corresponding TRel  $\wedge$  preorder TRel*  
 $\wedge \text{weak-reduction-correspondence-simulation TRel Target}$   
**using** *WOC-wrt-preorder-iff-reduction-correspondence-simulation[where TRel=TRel]*  
**by** *simp*  
**moreover** **have**  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$   
 $\wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}$   
 $\wedge \text{rel-reflects-divergence Rel (STCal Source Target)}$   
**apply** (*rule exI*) **using** *C1 C6 C7* **by** *blast*  
**hence** *enc-weakly-respects-barb-set {success}  $\wedge$  enc-reflects-divergence*  
**using** *WSS-DR-iff-source-target-rel*  
**by** *simp*  
**moreover** **have** *rel-weakly-respects-barb-set TRel TWB {success}*  
**proof** *auto*  
**fix** *TP TQ TP'*  
**assume**  $(TP, TQ) \in \text{TRel}$   
**with** *C2* **have**  $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$   
**by** *simp*  
**moreover** **assume**  $TP \longmapsto (\text{Calculus TWB}) * TP'$  **and**  $TP' \Downarrow \langle \text{TWB} \rangle \text{success}$   
**hence** *TargetTerm TP  $\Downarrow \langle \text{STCalWB SWB TWB} \rangle \text{success}$*   
**using** *STCalWB-reachesBarbST*

```

    by blast
ultimately have TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>success
  using C6
  by blast
thus TQ $\Downarrow$ <TWB>success
  using STCalWB-reachesBarbST
  by blast
next
fix TP TQ TQ'
assume (TP, TQ)  $\in$  TRel
with C2 have (TargetTerm TP, TargetTerm TQ)  $\in$  Rel
  by simp
moreover assume TQ  $\mapsto$  (Calculus TWB)* TQ' and TQ' $\Downarrow$ <TWB>success
hence TargetTerm TQ $\Downarrow$ <STCalWB SWB TWB>success
  using STCalWB-reachesBarbST
  by blast
ultimately have TargetTerm TP $\Downarrow$ <STCalWB SWB TWB>success
  using C6
  by blast
thus TP $\Downarrow$ <TWB>success
  using STCalWB-reachesBarbST
  by blast
qed
moreover from C2 C7 have rel-reflects-divergence TRel Target
  using STCal-divergent(2)
  by blast
ultimately
show weakly-operational-corresponding TRel  $\wedge$  preorder TRel
 $\wedge$  weak-reduction-correspondence-simulation TRel Target
 $\wedge$  enc-weakly-respects-barb-set {success}  $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
 $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence TRel Target
  by fast
qed

lemma (in encoding-wrt-barbs) OC-SS-DR-wrt-preorder-iff-source-target-rel:
fixes success :: 'barbs
and TRel :: ('procT  $\times$  'procT) set
shows (operational-corresponding TRel  $\wedge$  preorder TRel  $\wedge$  weak-reduction-bisimulation TRel Target
 $\wedge$  enc-weakly-respects-barb-set {success}
 $\wedge$  rel-weakly-respects-barb-set TRel TWB {success}
 $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence TRel Target)
= ( $\exists$  Rel. ( $\forall$  S. (SourceTerm S, TargetTerm ( $\llbracket S \rrbracket$ ))  $\in$  Rel)
 $\wedge$  TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2)  $\in$  Rel}
 $\wedge$  ( $\forall$  S T. (SourceTerm S, TargetTerm T)  $\in$  Rel  $\longrightarrow$  ( $\llbracket S \rrbracket$ , T)  $\in$  TRel)
 $\wedge$  weak-reduction-bisimulation Rel (STCal Source Target)  $\wedge$  preorder Rel
 $\wedge$  rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
 $\wedge$  rel-reflects-divergence Rel (STCal Source Target))
proof (rule iffI, (erule conjE)+)
assume A1: rel-weakly-preserves-barb-set TRel TWB {success}
and A2: rel-weakly-reflects-barb-set TRel TWB {success}
and A3: enc-weakly-preserves-barb-set {success}
and A4: enc-weakly-reflects-barb-set {success}
and A5: rel-reflects-divergence TRel Target and A6: enc-reflects-divergence
and A7: preorder TRel
from A7 have A8: TRel+ = TRel
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
from A7 have A9: TRel* = TRel
  using reflcl-trancl[of TRel] trancl-id[of TRel]
  unfolding preorder-on-def refl-on-def

```

by *auto*  
**define** *Rel* **where**  $Rel = indRelRTPO\ TRel$   
**hence**  $B1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
 by (*simp add: indRelRTPO.encR*)  
**from** *Rel-def A8* **have**  $B2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 using *indRelRTPO-to-TRel(4)*[**where**  $TRel=TRel$ ]  
 by (*auto simp add: indRelRTPO.target*)  
**from** *Rel-def A9* **have**  $B3: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$   
 using *indRelRTPO-to-TRel(2)*[**where**  $TRel=TRel$ ]  
*trans-closure-of-TRel-refl-cond*[**where**  $TRel=TRel$ ]  
 by *simp*  
**assume** *operational-complete TRel and operational-sound TRel and preorder TRel*  
**and** *weak-reduction-simulation TRel Target*  
**and**  $\forall P\ Q\ Q'. (P, Q) \in TRel \wedge Q \longmapsto Target* Q' \longrightarrow (\exists P'. P \longmapsto Target* P' \wedge (P', Q') \in TRel)$   
**with** *Rel-def A8 A9* **have**  $B4: weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$   
 using *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where**  $TRel=TRel$ ]  
 by *simp*  
**from** *Rel-def A7* **have**  $B5: preorder\ Rel$   
 using *indRelRTPO-is-preorder*[**where**  $TRel=TRel$ ]  
**unfolding** *preorder-on-def*  
 by *simp*  
**from** *Rel-def A1 A2 A3 A4* **have**  $B6: rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
 using *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*[**where**  $TRel=TRel$ ]  
**and** *success=success*  
 by *blast*  
**from** *Rel-def A5 A6* **have**  $B7: rel-reflects-divergence\ Rel\ (STCal\ Source\ Target)$   
 using *enc-and-TRelimpl-indRelRTPO-reflect-divergence*[**where**  $TRel=TRel$ ]  
 by *blast*  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target) \wedge preorder\ Rel$   
 $\wedge rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target)$   
**apply** (*rule exI*) **using**  $B1\ B2\ B3\ B4\ B5\ B6\ B7$  **by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target) \wedge preorder\ Rel$   
 $\wedge rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target)$   
**from** *this* **obtain** *Rel* **where**  $C1: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $C2: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3: \forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$   
**and**  $C4: weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$  **and**  $C5: preorder\ Rel$   
**and**  $C6: rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
**and**  $C7: rel-reflects-divergence\ Rel\ (STCal\ Source\ Target)$   
 by *auto*  
**from**  $C1\ C2\ C3\ C4\ C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge preorder\ Rel$   
 $\wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$   
 by *blast*  
**hence** *operational-corresponding TRel and preorder TRel and weak-reduction-bisimulation TRel Target*  
 using *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where**  $TRel=TRel$ ]  
 by *simp*  
**moreover** **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$   
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target)$   
**apply** (*rule exI*) **using**  $C1\ C6\ C7$  **by** *blast*

**hence** *enc-weakly-respects-barb-set* {*success*}  $\wedge$  *enc-reflects-divergence*  
**using** *WSS-DR-iff-source-target-rel*  
**by** *simp*  
**moreover have** *rel-weakly-respects-barb-set* *TRel TWB* {*success*}  
**proof auto**  
**fix** *TP TQ TP'*  
**assume**  $(TP, TQ) \in TRel$   
**with** *C2* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TP \mapsto (Calculus\ TWB)^* TP'$  **and**  $TP' \Downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $TargetTerm\ TQ \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *C6*  
**by** *blast*  
**thus**  $TQ \Downarrow \langle TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**next**  
**fix** *TP TQ TQ'*  
**assume**  $(TP, TQ) \in TRel$   
**with** *C2* **have**  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$   
**by** *simp*  
**moreover assume**  $TQ \mapsto (Calculus\ TWB)^* TQ'$  **and**  $TQ' \Downarrow \langle TWB \rangle success$   
**hence**  $TargetTerm\ TQ \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**ultimately have**  $TargetTerm\ TP \Downarrow \langle STCalWB\ SWB\ TWB \rangle success$   
**using** *C6*  
**by** *blast*  
**thus**  $TP \Downarrow \langle TWB \rangle success$   
**using** *STCalWB-reachesBarbST*  
**by** *blast*  
**qed**  
**moreover from** *C2 C7* **have** *rel-reflects-divergence* *TRel Target*  
**using** *STCal-divergent(2)*  
**by** *blast*  
**ultimately**  
**show** *operational-corresponding* *TRel*  $\wedge$  *preorder* *TRel*  $\wedge$  *weak-reduction-bisimulation* *TRel Target*  
 $\wedge$  *enc-weakly-respects-barb-set* {*success*}  $\wedge$  *rel-weakly-respects-barb-set* *TRel TWB* {*success*}  
 $\wedge$  *enc-reflects-divergence*  $\wedge$  *rel-reflects-divergence* *TRel Target*  
**by** *fast*  
**qed**

**lemma** (in *encoding-wrt-barbs*) *SOC-SS-DR-wrt-preorder-iff-source-target-rel*:

**fixes** *success* :: 'barbs

**and** *TRel* :: ('procT  $\times$  'procT) set

**shows** (*strongly-operational-corresponding* *TRel*  $\wedge$  *preorder* *TRel*

$\wedge$  *strong-reduction-bisimulation* *TRel Target*

$\wedge$  *enc-respects-barb-set* {*success*}  $\wedge$  *rel-respects-barb-set* *TRel TWB* {*success*}

$\wedge$  *enc-reflects-divergence*  $\wedge$  *rel-reflects-divergence* *TRel Target*)

=  $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$

$\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$

$\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$

$\wedge$  *strong-reduction-bisimulation* *Rel* (*STCal* *Source Target*)  $\wedge$  *preorder* *Rel*

$\wedge$  *rel-respects-barb-set* *Rel* (*STCalWB* *SWB* *TWB*) {*success*}

$\wedge$  *rel-reflects-divergence* *Rel* (*STCal* *Source Target*)

**proof** (*rule iffI*, (*erule conjE*) $+$ )

**assume** *A1*: *rel-preserves-barb-set* *TRel TWB* {*success*}

**and** *A2*: *rel-reflects-barb-set* *TRel TWB* {*success*}

**and**  $A3$ : *enc-preserves-barb-set*  $\{success\}$  **and**  $A4$ : *enc-reflects-barb-set*  $\{success\}$   
**and**  $A5$ : *rel-reflects-divergence*  $TRel$  **Target** **and**  $A6$ : *enc-reflects-divergence*  
**and**  $A7$ : *preorder*  $TRel$   
**from**  $A7$  **have**  $A8$ :  $TRel^+ = TRel$   
**using** *trancl-id*[of  $TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *blast*  
**from**  $A7$  **have**  $A9$ :  $TRel^* = TRel$   
**using** *reflcl-trancl*[of  $TRel$ ] *trancl-id*[of  $TRel$ ]  
**unfolding** *preorder-on-def* *refl-on-def*  
**by** *auto*  
**define**  $Rel$  **where**  $Rel = indRelRTPO$   $TRel$   
**hence**  $B1$ :  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**by** (*simp add: indRelRTPO.encR*)  
**from** *Rel-def*  $A8$  **have**  $B2$ :  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**using** *indRelRTPO-to-TRel(4)*[**where**  $TRel = TRel$ ]  
**by** (*auto simp add: indRelRTPO.target*)  
**from** *Rel-def*  $A9$  **have**  $B3$ :  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$   
**using** *indRelRTPO-to-TRel(2)*[**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**assume** *strongly-operational-complete*  $TRel$  **and** *strongly-operational-sound*  $TRel$   
**and** *preorder*  $TRel$  **and** *strong-reduction-simulation*  $TRel$  **Target**  
**and**  $\forall P\ Q\ Q'. (P, Q) \in TRel \wedge Q \longmapsto Target\ Q' \longrightarrow (\exists P'. P \longmapsto Target\ P' \wedge (P', Q') \in TRel)$   
**with** *Rel-def*  $A8$   $A9$  **have**  $B4$ : *strong-reduction-bisimulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**using** *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *simp*  
**from** *Rel-def*  $A7$  **have**  $B5$ : *preorder*  $Rel$   
**using** *indRelRTPO-is-preorder*[**where**  $TRel = TRel$ ]  
**unfolding** *preorder-on-def*  
**by** *simp*  
**from** *Rel-def*  $A1$   $A2$   $A3$   $A4$  **have**  $B6$ : *rel-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**using** *enc-and-TRel-impl-indRelRTPO-respects-success*[**where**  $TRel = TRel$  **and**  $success = success$ ]  
**by** *blast*  
**from** *Rel-def*  $A5$   $A6$  **have**  $B7$ : *rel-reflects-divergence*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**using** *enc-and-TRelimpl-indRelRTPO-reflect-divergence*[**where**  $TRel = TRel$ ]  
**by** *blast*  
**show**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *strong-reduction-bisimulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
 $\wedge$  *rel-reflects-divergence*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**apply** (*rule exI*) **using**  $B1$   $B2$   $B3$   $B4$   $B5$   $B6$   $B7$  **by** *blast*  
**next**  
**assume**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$   
 $\wedge$  *strong-reduction-bisimulation*  $Rel$  ( $STCal$   $Source$   $Target$ )  $\wedge$  *preorder*  $Rel$   
 $\wedge$  *rel-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
 $\wedge$  *rel-reflects-divergence*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**from** *this* **obtain**  $Rel$  **where**  $C1$ :  $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$   
**and**  $C2$ :  $TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
**and**  $C3$ :  $\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel$   
**and**  $C4$ : *strong-reduction-bisimulation*  $Rel$  ( $STCal$   $Source$   $Target$ ) **and**  $C5$ : *preorder*  $Rel$   
**and**  $C6$ : *rel-respects-barb-set*  $Rel$  ( $STCalWB$   $SWB$   $TWB$ )  $\{success\}$   
**and**  $C7$ : *rel-reflects-divergence*  $Rel$  ( $STCal$   $Source$   $Target$ )  
**by** *auto*  
**from**  $C1$   $C2$   $C3$   $C4$   $C5$  **have**  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$   
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$   
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \wedge$  *preorder*  $Rel$

```

 $\wedge$  strong-reduction-bisimulation  $Rel$  ( $STCal$   $Source$   $Target$ )
  by blast
hence strongly-operational-corresponding  $TRel$   $\wedge$  preorder  $TRel$ 
   $\wedge$  strong-reduction-bisimulation  $TRel$   $Target$ 
  using SOC-wrt-preorder-iff-strong-reduction-bisimulation[where  $TRel=TRel$ ]
  by simp
moreover have  $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$ 
   $\wedge$  rel-respects-barb-set  $Rel$  ( $STCalWB$   $SWB$   $TWB$ ) {success}
   $\wedge$  rel-reflects-divergence  $Rel$  ( $STCal$   $Source$   $Target$ )
  apply (rule exI) using C1 C6 C7 by blast
hence enc-respects-barb-set {success}  $\wedge$  enc-reflects-divergence
  using SS-DR-iff-source-target-rel
  by simp
moreover have rel-respects-barb-set  $TRel$   $TWB$  {success}
proof auto
  fix  $TP$   $TQ$ 
  assume  $(TP, TQ) \in TRel$ 
  with C2 have  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$ 
    by simp
  moreover assume  $TP \downarrow_{\langle TWB \rangle} success$ 
  hence  $TargetTerm\ TP \downarrow_{\langle STCalWB\ SWB\ TWB \rangle} success$ 
    using  $STCalWB$ -hasBarbST
    by blast
  ultimately have  $TargetTerm\ TQ \downarrow_{\langle STCalWB\ SWB\ TWB \rangle} success$ 
    using C6
    by blast
  thus  $TQ \downarrow_{\langle TWB \rangle} success$ 
    using  $STCalWB$ -hasBarbST
    by blast
next
  fix  $TP$   $TQ$ 
  assume  $(TP, TQ) \in TRel$ 
  with C2 have  $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$ 
    by simp
  moreover assume  $TQ \downarrow_{\langle TWB \rangle} success$ 
  hence  $TargetTerm\ TQ \downarrow_{\langle STCalWB\ SWB\ TWB \rangle} success$ 
    using  $STCalWB$ -hasBarbST
    by blast
  ultimately have  $TargetTerm\ TP \downarrow_{\langle STCalWB\ SWB\ TWB \rangle} success$ 
    using C6
    by blast
  thus  $TP \downarrow_{\langle TWB \rangle} success$ 
    using  $STCalWB$ -hasBarbST
    by blast
qed
moreover from C2 C7 have rel-reflects-divergence  $TRel$   $Target$ 
  using  $STCal$ -divergent(2)
  by blast
ultimately show strongly-operational-corresponding  $TRel$   $\wedge$  preorder  $TRel$ 
   $\wedge$  strong-reduction-bisimulation  $TRel$   $Target$ 
   $\wedge$  enc-respects-barb-set {success}  $\wedge$  rel-respects-barb-set  $TRel$   $TWB$  {success}
   $\wedge$  enc-reflects-divergence  $\wedge$  rel-reflects-divergence  $TRel$   $Target$ 
  by fast
qed

```

### 10.3 Full Abstraction and Operational Correspondence

To combine full abstraction and operational correspondence we consider a symmetric version of the induced relation and assume that the relations  $SRel$  and  $TRel$  are equivalences. Then an encoding is fully abstract w.r.t.  $SRel$  and  $TRel$  and operationally corresponding w.r.t.  $TRel$  such that  $TRel$  is a

bisimulation iff the induced relation contains both SRel and TRel and is a transitive bisimulation.

**lemma** (in *encoding*) *FS-OC-modulo-equivalences-iff-source-target-relation*:

**fixes**  $SRel :: ('procS \times 'procS)$  set

**and**  $TRel :: ('procT \times 'procT)$  set

**assumes**  $eqS$ : equivalence SRel

**and**  $eqT$ : equivalence TRel

**shows** *fully-abstract SRel TRel*

$\wedge$  *operational-corresponding TRel*  $\wedge$  *weak-reduction-bisimulation TRel Target*

$\longleftrightarrow (\exists Rel.$

$(\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$

$\wedge trans\ Rel \wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target))$

**proof** (rule *iffI*, erule *conjE*, erule *conjE*)

**assume**  $A1$ : *fully-abstract SRel TRel* **and**  $A2$ : *operational-corresponding TRel*

**and**  $A3$ : *weak-reduction-bisimulation TRel Target*

**from**  $eqT$  **have**  $A4$ :  $TRel^* = TRel$

**using** *reflcl-trancl*[of TRel] *trancl-id*[of TRel]

**unfolding** *equiv-def refl-on-def*

**by** *auto*

**have**  $A5$ :

$\forall S. SourceTerm S \sim[\cdot]T < TRel > TargetTerm (\llbracket S \rrbracket) \wedge TargetTerm (\llbracket S \rrbracket) \sim[\cdot]T < TRel > SourceTerm S$

**by** (*simp add: indRelTEQ.encR indRelTEQ.encL*)

**moreover from**  $A4$  **have**  $A6$ :  $TRel = \{(T1, T2). TargetTerm T1 \sim[\cdot]T < TRel > TargetTerm T2\}$

**using** *indRelTEQ-to-TRel(4)*[**where**  $TRel = TRel$ ]

*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]

**by** (*auto simp add: indRelTEQ.target*)

**moreover have**  $A7$ : *trans (indRelTEQ TRel)*

**using** *indRelTEQ.trans*[**where**  $TRel = TRel$ ]

**unfolding** *trans-def*

**by** *blast*

**moreover have**  $SRel = \{(S1, S2). SourceTerm S1 \sim[\cdot]T < TRel > SourceTerm S2\}$

**proof** –

**from**  $A6$  **have**  $\forall S1\ S2. ((\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in TRel) = TargetTerm (\llbracket S1 \rrbracket) \sim[\cdot]T < TRel > TargetTerm (\llbracket S2 \rrbracket)$

**by** *blast*

**moreover have**  $indRelTEQ\ TRel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\} = indRelTEQ\ TRel$

**by** (*auto simp add: indRelTEQ.encL*)

**with**  $A7$  **have** *trans (indRelTEQ TRel  $\cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\})$*

**unfolding** *trans-def*

**by** *blast*

**ultimately show**  $SRel = \{(S1, S2). SourceTerm S1 \sim[\cdot]T < TRel > SourceTerm S2\}$

**using**  $A1\ A5$  *full-abstraction-and-trans-relation-contains-TRel-impl-SRel*[**where**

$SRel = SRel$  **and**  $TRel = TRel$  **and**  $Rel = indRelTEQ\ TRel$ ]

**by** *blast*

**qed**

**moreover from**  $eqT\ A2\ A3$  **have** *weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)*

**using** *OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation*[**where**  $TRel = TRel$ ]

**by** *blast*

**ultimately**

**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$

$\wedge trans\ Rel \wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$

**by** *blast*

**next**

**assume**

$\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$

$\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$

$\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$

$\wedge trans\ Rel \wedge weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$



from *this* obtain *Rel* where

*B1*:  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel} \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel}$

and *B2*:  $S\text{Rel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$

and *B3*:  $T\text{Rel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

and *B4*: *trans Rel* and *B5*: *weak-reduction-bisimulation Rel* (*STCal Source Target*)

by *blast*

from *B1 B2 B3 B4* have *fully-abstract SRel TRel*

using *trans-source-target-relation-impl-fully-abstract*[**where** *Rel=Rel* and *SRel=SRel*

and *TRel=TRel*]

by *blast*

moreover have *operational-corresponding TRel*  $\wedge$  *weak-reduction-bisimulation TRel Target*

**proof** –

from *eqT* have *C1*:  $T\text{Rel}^+ = T\text{Rel}$

using *trancl-id*[of *TRel*]

unfolding *equiv-def*

by *blast*

from *eqT* have *C2*:  $T\text{Rel}^* = T\text{Rel}$

using *reflct-trancl*[of *TRel*] *trancl-id*[of *TRel*]

unfolding *equiv-def refl-on-def*

by *auto*

from *B1* have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

by *simp*

moreover from *B3* have  $\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$

by *simp*

moreover from *B3 C1* have  $\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+$

by *simp*

moreover have  $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*$

**proof** *clarify*

fix *S T*

from *B1* have  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel}$

by *simp*

moreover assume  $(\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}$

ultimately have  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}$

using *B4*

unfolding *trans-def*

by *blast*

with *B3 C2* show  $(\llbracket S \rrbracket, T) \in T\text{Rel}^*$

by *simp*

**qed**

ultimately have  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge (\forall T1 T2. (T1, T2) \in T\text{Rel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in T\text{Rel}^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in T\text{Rel}^*)$

$\wedge \text{weak-reduction-bisimulation Rel } (\text{STCal Source Target})$

using *B5*

by *blast*

with *C1 C2* show *operational-corresponding TRel*  $\wedge$  *weak-reduction-bisimulation TRel Target*

using *OC-iff-weak-reduction-bisimulation*[**where** *TRel=TRel*]

by *auto*

**qed**

ultimately show *fully-abstract SRel TRel*  $\wedge$  *operational-corresponding TRel*

$\wedge$  *weak-reduction-bisimulation TRel Target*

by *simp*

**qed**

**lemma** (*in encoding*) *FA-SOC-modulo-equivalences-iff-source-target-relation*:

fixes *SRel* :: ('procS  $\times$  'procS) set

and *TRel* :: ('procT  $\times$  'procT) set

assumes *eqS*: *equivalence SRel*

and *eqT*: *equivalence TRel*

shows *fully-abstract SRel TRel*  $\wedge$  *strongly-operational-corresponding TRel*

$\wedge$  strong-reduction-bisimulation  $TRel$   $Target \longleftrightarrow (\exists Rel.$   
 $(\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \wedge (TargetTerm ([S]), SourceTerm S) \in Rel)$   
 $\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$   
 $\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} \wedge trans Rel$   
 $\wedge$  strong-reduction-bisimulation  $Rel$  ( $STCal$   $Source$   $Target$ )

**proof** (rule *iffI*, erule *conjE*, erule *conjE*)

**assume**  $A1$ : fully-abstract  $SRel$   $TRel$  **and**  $A2$ : strongly-operational-corresponding  $TRel$   
**and**  $A3$ : strong-reduction-bisimulation  $TRel$   $Target$

**from**  $eqT$  **have**  $A4$ :  $TRel^* = TRel$   
**using** *reflcl-trancl*[of  $TRel$ ] *trancl-id*[of  $TRel$ ]  
**unfolding** *equiv-def refl-on-def*  
**by** *auto*

**have**  $A5$ :  
 $\forall S. SourceTerm S \sim[\cdot]T < TRel > TargetTerm ([S]) \wedge TargetTerm ([S]) \sim[\cdot]T < TRel > SourceTerm S$   
**by** (*simp add: indRelTEQ.encR indRelTEQ.encL*)

**moreover from**  $A4$  **have**  $A6$ :  $TRel = \{(T1, T2). TargetTerm T1 \sim[\cdot]T < TRel > TargetTerm T2\}$   
**using** *indRelTEQ-to-TRel*(4)[**where**  $TRel = TRel$ ]  
*trans-closure-of-TRel-refl-cond*[**where**  $TRel = TRel$ ]  
**by** (*auto simp add: indRelTEQ.target*)

**moreover have**  $A7$ : *trans* (*indRelTEQ*  $TRel$ )  
**using** *indRelTEQ.trans*[**where**  $TRel = TRel$ ]  
**unfolding** *trans-def*  
**by** *blast*

**moreover have**  $SRel = \{(S1, S2). SourceTerm S1 \sim[\cdot]T < TRel > SourceTerm S2\}$

**proof** –

**from**  $A6$  **have**  $\forall S1 S2. (([S1], [S2]) \in TRel) = TargetTerm ([S1]) \sim[\cdot]T < TRel > TargetTerm ([S2])$   
**by** *blast*

**moreover have** *indRelTEQ*  $TRel \cup \{(P, Q). \exists S. [S] \in T P \wedge S \in S Q\} = indRelTEQ TRel$   
**by** (*auto simp add: indRelTEQ.encL*)

**with**  $A7$  **have** *trans* (*indRelTEQ*  $TRel \cup \{(P, Q). \exists S. [S] \in T P \wedge S \in S Q\}$ )  
**unfolding** *trans-def*  
**by** *blast*

**ultimately show**  $SRel = \{(S1, S2). SourceTerm S1 \sim[\cdot]T < TRel > SourceTerm S2\}$   
**using**  $A1$   $A5$  *full-abstraction-and-trans-relation-contains-TRel-impl-SRel*[**where**  
 $SRel = SRel$  **and**  $TRel = TRel$  **and**  $Rel = indRelTEQ TRel$ ]

**by** *blast*

**qed**

**moreover from**  $eqT$   $A2$   $A3$  **have** strong-reduction-bisimulation (*indRelTEQ*  $TRel$ ) ( $STCal$   $Source$   $Target$ )  
**using** *SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation*[**where**  $TRel = TRel$ ]  
**by** *blast*

**ultimately**

**show**  $\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \wedge (TargetTerm ([S]), SourceTerm S) \in Rel)$   
 $\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$   
 $\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} \wedge trans Rel$   
 $\wedge$  strong-reduction-bisimulation  $Rel$  ( $STCal$   $Source$   $Target$ )

**by** *blast*

**next**

**assume**  
 $\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \wedge (TargetTerm ([S]), SourceTerm S) \in Rel)$   
 $\wedge SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$   
 $\wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} \wedge trans Rel$   
 $\wedge$  strong-reduction-bisimulation  $Rel$  ( $STCal$   $Source$   $Target$ )

**from this obtain**  $Rel$  **where**  
 $B1$ :  $\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \wedge (TargetTerm ([S]), SourceTerm S) \in Rel$   
**and**  $B2$ :  $SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\}$   
**and**  $B3$ :  $TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$  **and**  $B4$ : *trans*  $Rel$   
**and**  $B5$ : strong-reduction-bisimulation  $Rel$  ( $STCal$   $Source$   $Target$ )

**by** *blast*

**from**  $B1$   $B2$   $B3$   $B4$  **have** fully-abstract  $SRel$   $TRel$   
**using** *trans-source-target-relation-impl-fully-abstract*[**where**  $Rel = Rel$  **and**  $SRel = SRel$   
**and**  $TRel = TRel$ ]

```

  by blast
moreover
have strongly-operational-corresponding TRel  $\wedge$  strong-reduction-bisimulation TRel Target
proof -
  from eqT have C1: TRel+ = TRel
    using trancl-id[of TRel]
    unfolding equiv-def refl-on-def
  by blast
  from eqT have C2: TRel* = TRel
    using reflcl-trancl[of TRel] trancl-id[of TRel]
    unfolding equiv-def refl-on-def
  by auto
  from B1 have  $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$ 
  by simp
  moreover from B3 have  $\forall T1\ T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$ 
  by simp
  moreover from B3 C1
  have  $\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+$ 
  by simp
  moreover have  $\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$ 
proof clarify
  fix S T
  from B1 have  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel}$ 
  by simp
  moreover assume  $(\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}$ 
  ultimately have  $(\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}$ 
    using B4
    unfolding trans-def
  by blast
  with B3 C2 show  $(\llbracket S \rrbracket, T) \in \text{TRel}^*$ 
  by simp
qed
ultimately have  $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$ 
 $\wedge (\forall T1\ T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$ 
 $\wedge (\forall T1\ T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$ 
 $\wedge (\forall S\ T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$ 
 $\wedge \text{strong-reduction-bisimulation Rel } (\text{STCal Source Target})$ 
  using B5
  by blast
with C1 C2
show strongly-operational-corresponding TRel  $\wedge$  strong-reduction-bisimulation TRel Target
  using SOC-iff-strong-reduction-bisimulation[where TRel=TRel]
  by auto
qed
ultimately show fully-abstract SRel TRel  $\wedge$  strongly-operational-corresponding TRel
   $\wedge$  strong-reduction-bisimulation TRel Target
  by simp
qed

```

An encoding that is fully abstract w.r.t. the equivalences SRel and TRel and operationally corresponding w.r.t. TRel ensures that SRel is a bisimulation iff TRel is a bisimulation.

**lemma** (in *encoding*) *FA-and-OC-and-TRel-impl-SRel-bisimulation:*

```

fixes SRel :: ('procS  $\times$  'procS) set
and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
and opCom: operational-complete TRel
and opSou: operational-sound TRel
and symmT: sym TRel
and transT: trans TRel
and bisimT: weak-reduction-bisimulation TRel Target

```

shows *weak-reduction-bisimulation SRel Source*

```

proof auto
  fix SP SQ SP'
  assume SP  $\mapsto$ Source* SP'
  with opCom obtain TP' where A1:  $\llbracket SP \rrbracket \mapsto$ Target* TP' and A2:  $(\llbracket SP \rrbracket, TP') \in TRel$ 
    by blast
  assume (SP, SQ)  $\in SRel$ 
  with fullAbs have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    by simp
  with bisimT A1 obtain TQ' where A3:  $\llbracket SQ \rrbracket \mapsto$ Target* TQ' and A4:  $(TP', TQ') \in TRel$ 
    by blast
  from A3 opSou obtain SQ' where A5: SQ  $\mapsto$ Source* SQ' and A6:  $(\llbracket SQ \rrbracket, TQ') \in TRel$ 
    by blast
  from A2 A4 A6 symmT transT have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    unfolding trans-def sym-def
    by blast
  with fullAbs A5 show  $\exists SQ'. SQ \mapsto$ Source* SQ'  $\wedge (SP', SQ') \in SRel$ 
    by blast
next
  fix SP SQ SQ'
  assume SQ  $\mapsto$ Source* SQ'
  with opCom obtain TQ' where B1:  $\llbracket SQ \rrbracket \mapsto$ Target* TQ' and B2:  $(\llbracket SQ \rrbracket, TQ') \in TRel$ 
    by blast
  assume (SP, SQ)  $\in SRel$ 
  with fullAbs have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    by simp
  with bisimT B1 obtain TP' where B3:  $\llbracket SP \rrbracket \mapsto$ Target* TP' and B4:  $(TP', TQ') \in TRel$ 
    by blast
  from B3 opSou obtain SP' where B5: SP  $\mapsto$ Source* SP' and B6:  $(\llbracket SP \rrbracket, TP') \in TRel$ 
    by blast
  from B2 B4 B6 symmT transT have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    unfolding trans-def sym-def
    by blast
  with fullAbs B5 show  $\exists SP'. SP \mapsto$ Source* SP'  $\wedge (SP', SQ') \in SRel$ 
    by blast
qed

```

**lemma** (in *encoding*) *FA-and-SOC-and-TRel-impl-SRel-strong-bisimulation:*

```

fixes SRel :: ('procS  $\times$  'procS) set
  and TRel :: ('procT  $\times$  'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and opCom: strongly-operational-complete TRel
  and opSou: strongly-operational-sound TRel
  and symmT: sym TRel
  and transT: trans TRel
  and bisimT: strong-reduction-bisimulation TRel Target
shows strong-reduction-bisimulation SRel Source
proof auto
  fix SP SQ SP'
  assume SP  $\mapsto$ Source SP'
  with opCom obtain TP' where A1:  $\llbracket SP \rrbracket \mapsto$ Target TP' and A2:  $(\llbracket SP \rrbracket, TP') \in TRel$ 
    by blast
  assume (SP, SQ)  $\in SRel$ 
  with fullAbs have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    by simp
  with bisimT A1 obtain TQ' where A3:  $\llbracket SQ \rrbracket \mapsto$ Target TQ' and A4:  $(TP', TQ') \in TRel$ 
    by blast
  from A3 opSou obtain SQ' where A5: SQ  $\mapsto$ Source SQ' and A6:  $(\llbracket SQ \rrbracket, TQ') \in TRel$ 
    by blast
  from A2 A4 A6 symmT transT have  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$ 
    unfolding trans-def sym-def

```

by *blast*  
 with *fullAbs*  $A5$  show  $\exists SQ'. SQ \mapsto_{Source} SQ' \wedge (SP', SQ') \in SRel$   
 by *blast*  
**next**  
 fix  $SP\ SQ\ SQ'$   
 assume  $SQ \mapsto_{Source} SQ'$   
 with *opCom* **obtain**  $TQ'$  where  $B1: \llbracket SQ \rrbracket \mapsto_{Target} TQ'$  and  $B2: (\llbracket SQ \rrbracket, TQ') \in TRel$   
 by *blast*  
 assume  $(SP, SQ) \in SRel$   
 with *fullAbs* **have**  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
 by *simp*  
 with *bisimT*  $B1$  **obtain**  $TP'$  where  $B3: \llbracket SP \rrbracket \mapsto_{Target} TP'$  and  $B4: (TP', TQ') \in TRel$   
 by *blast*  
 from  $B3$  *opSou* **obtain**  $SP'$  where  $B5: SP \mapsto_{Source} SP'$  and  $B6: (\llbracket SP \rrbracket, TP') \in TRel$   
 by *blast*  
 from  $B2\ B4\ B6$  *symmT transT* **have**  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
 unfolding *trans-def sym-def*  
 by *blast*  
 with *fullAbs*  $B5$  show  $\exists SP'. SP \mapsto_{Source} SP' \wedge (SP', SQ') \in SRel$   
 by *blast*  
**qed**

**lemma** (in *encoding*) *FA-and-OC-impl-SRel-iff-TRel-bisimulation*:

fixes  $SRel :: ('procS \times 'procS)$  set  
 and  $TRel :: ('procT \times 'procT)$  set  
 assumes *fullAbs*: *fully-abstract*  $SRel\ TRel$   
 and *opCor*: *operational-corresponding*  $TRel$   
 and *symmT*: *sym*  $TRel$   
 and *transT*: *trans*  $TRel$   
 and *surj*:  $\forall T. \exists S. T = \llbracket S \rrbracket$   
 shows *weak-reduction-bisimulation*  $SRel\ Source \longleftrightarrow$  *weak-reduction-bisimulation*  $TRel\ Target$   
**proof**  
 assume *bisimS*: *weak-reduction-bisimulation*  $SRel\ Source$   
 have *weak-reduction-simulation*  $TRel\ Target$   
**proof** *clarify*  
 fix  $TP\ TQ\ TP'$   
 from *surj* **have**  $\exists S. TP = \llbracket S \rrbracket$   
 by *simp*  
 from *this* **obtain**  $SP$  where  $A1: \llbracket SP \rrbracket = TP$   
 by *blast*  
 from *surj* **have**  $\exists S. TQ = \llbracket S \rrbracket$   
 by *simp*  
 from *this* **obtain**  $SQ$  where  $A2: \llbracket SQ \rrbracket = TQ$   
 by *blast*  
 assume  $TP \mapsto_{Target*} TP'$   
 with *opCor*  $A1$  **obtain**  $SP'$  where  $A3: SP \mapsto_{Source*} SP'$  and  $A4: (\llbracket SP \rrbracket, TP') \in TRel$   
 by *blast*  
 assume  $(TP, TQ) \in TRel$   
 with *fullAbs*  $A1\ A2$  **have**  $(SP, SQ) \in SRel$   
 by *simp*  
 with *bisimS*  $A3$  **obtain**  $SQ'$  where  $A5: SQ \mapsto_{Source*} SQ'$  and  $A6: (SP', SQ') \in SRel$   
 by *blast*  
 from *opCor*  $A2\ A5$  **obtain**  $TQ'$  where  $A7: TQ \mapsto_{Target*} TQ'$  and  $A8: (\llbracket SQ \rrbracket, TQ') \in TRel$   
 by *blast*  
 from *symmT*  $A4$  **have**  $(TP', \llbracket SP \rrbracket) \in TRel$   
 unfolding *sym-def*  
 by *simp*  
 moreover from *fullAbs*  $A6$  **have**  $(\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$   
 by *simp*  
 ultimately **have**  $(TP', TQ') \in TRel$   
 using *transT*  $A8$

```

    unfolding trans-def
  by blast
with A7 show  $\exists TQ'. TQ \mapsto_{Target*} TQ' \wedge (TP', TQ') \in TRel$ 
  by blast
qed
with symmT show weak-reduction-bisimulation TRel Target
  using symm-weak-reduction-simulation-is-bisimulation[where Rel=TRel and Cal=Target]
  by blast
next
  assume weak-reduction-bisimulation TRel Target
with fullAbs opCor symmT transT show weak-reduction-bisimulation SRel Source
  using FA-and-OC-and-TRel-impl-SRel-bisimulation[where SRel=SRel and TRel=TRel]
  by blast
qed
end

```