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

Kirstin Peters*
TU Dresden, Germany

Rob van Glabbeek
NICTA,† 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').

†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

1 Relations **3**
1.1 Basic Conditions . . . 3
1.2 Preservation, Reflection, and Respection of Predicates . . . 6

2 Process Calculi **11**
2.1 Reduction Semantics . . . 11
2.1.1 Observables or Barbs . . . 13

3 Simulation Relations **17**
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

4 Encodings **46**

5 Relation between Source and Target Terms **54**
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

6 Success Sensitiveness and Barbs **151**

7 Divergence Reflection **155**

8 Operational Correspondence **156**
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

9 Full Abstraction **201**
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

10 Combining Criteria **235**
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, preoders, equivalence, and inverse relations.

abbreviation *preorder Rel* ≡ *preorder-on UNIV Rel*
abbreviation *equivalence Rel* ≡ *equiv UNIV Rel*

A symmetric preorder is an equivalence.

lemma *symm-preorder-is-equivalence*:
fixes *Rel* :: $('a \times 'a)$ *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* :: $('a \times 'a)$ *set* ⇒ $('a \times 'a)$ *set* **where**
symcl Rel = *Rel* ∪ Rel^{-1}

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* :: $('a \times 'a)$ *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* :: $('a \times 'a)$ *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* :: $('a \times 'a)$ *set*
shows *symcl* ($Rel^{=}$) = (*symcl Rel*)$^{=}$
by (*auto simp add*: *symcl-def refl*)

The symmetric closure of a reflexive relation is reflexive.

lemma *refl-symcl-of-refl-rel*:
fixes *Rel* :: $('a \times 'a)$ *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* :: ($'a \times 'a$) *set*
  shows (*symcl* ($Rel^{=}$))$^{+}$ = (*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* :: ($'a \times 'a$) *set*
  assumes *sym Rel*
  shows *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* :: ($'a \times 'a$) *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* :: ($'a \times 'a$) *set*
  assumes *sym Rel*
  shows *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* :: ($'a \times 'a$) *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 transtive closure.

lemma *refl-trans-closure-subset-of-refl-symm-trans-closure*:
  fixes *Rel* :: ($'a \times 'a$) *set*
  shows $Rel^{*} \subseteq$ (*symcl* ($Rel^{=}$))$^{+}$
proof *clarify*
  fix *a b*
  assume (a, b) $\in Rel^{*}$
  hence (a, b) $\in$ (*symcl Rel*)*
    using *in-rtrancl-UnI*[*of* (a, b) *Rel* Rel^{-1}]
   by (*simp add*: *symcl-def*)
  thus (a, b) $\in$ (*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 × 'a) set
  assumes condA: ∀ a b c. (a, b) ∈ Rel ∧ (c, b) ∈ Rel ∧ (a, c) ∉ Rel
                 ⟶ (b, a) ∈ Rel ∨ (b, c) ∈ Rel
    and condB: ∀ a b c. (a, b) ∈ Rel ∧ (a, c) ∈ Rel ∧ (b, c) ∉ Rel
                 ⟶ (b, a) ∈ Rel ∨ (c, a) ∈ Rel
    and reflR: refl Rel
    and tranR: trans Rel
  shows trans (symcl Rel)
   unfolding trans-def
proof clarify
  fix a b c
  have ⟦(a, b) ∈ Rel; (b, c) ∈ Rel⟧ ⟹ (a, c) ∈ symcl Rel
  proof −
    assume (a, b) ∈ Rel and (b, c) ∈ Rel
    with tranR have (a, c) ∈ Rel
        unfolding trans-def
      by blast
    thus (a, c) ∈ symcl Rel
      by (simp add: symcl-def)
  qed
  moreover have ⟦(a, b) ∈ Rel; (c, b) ∈ Rel; (a, c) ∉ Rel⟧ ⟹ (a, c) ∈ symcl Rel
  proof −
    assume A1: (a, b) ∈ Rel and A2: (c, b) ∈ Rel and (a, c) ∉ Rel
    with condA have (b, a) ∈ Rel ∨ (b, c) ∈ Rel
      by blast
    thus (a, c) ∈ symcl Rel
    proof auto
      assume (b, a) ∈ Rel
      with A2 tranR have (c, a) ∈ Rel
          unfolding trans-def
        by blast
      thus (a, c) ∈ symcl Rel
        by (simp add: symcl-def)
    next
      assume (b, c) ∈ Rel
      with A1 tranR have (a, c) ∈ Rel
          unfolding trans-def
        by blast
      thus (a, c) ∈ symcl Rel
        by (simp add: symcl-def)
    qed
  qed
  moreover have ⟦(b, a) ∈ Rel; (b, c) ∈ Rel; (a, c) ∉ Rel⟧ ⟹ (a, c) ∈ symcl Rel
  proof −
    assume B1: (b, a) ∈ Rel and B2: (b, c) ∈ Rel and (a, c) ∉ Rel
    with condB have (a, b) ∈ Rel ∨ (c, b) ∈ Rel
      by blast
    thus (a, c) ∈ symcl Rel
    proof auto
      assume (a, b) ∈ Rel
      with B2 tranR have (a, c) ∈ Rel
          unfolding trans-def
        by blast
      thus (a, c) ∈ symcl Rel
        by (simp add: symcl-def)
    next
      assume (c, b) ∈ Rel
      with B1 tranR have (c, a) ∈ Rel
          unfolding trans-def
        by blast
```

thus $(a, c) \in$ *symcl Rel*
by (*simp add*: *symcl-def*)
qed
qed
moreover have $[\![(b, a) \in Rel; (c, b) \in Rel]\!] \Longrightarrow (a, c) \in$ *symcl Rel*
proof −
assume $(c, b) \in Rel$ **and** $(b, a) \in Rel$
with *tranR* **have** $(c, a) \in Rel$
unfolding *trans-def*
by *blast*
thus $(a, c) \in$ *symcl Rel*
by (*simp add*: *symcl-def*)
qed
moreover assume $(a, b) \in$ *symcl Rel* **and** $(b, c) \in$ *symcl Rel*
ultimately show $(a, c) \in$ *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-preserves-pred* :: $('a \times 'a)$ *set* $\Rightarrow ('a \Rightarrow bool) \Rightarrow bool$ **where**
rel-preserves-pred Rel Pred $\equiv \forall\, a\ b.\ (a, b) \in Rel \wedge Pred\ a \longrightarrow Pred\ b$

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

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

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

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

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

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

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

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

lemma *symm-relation-impl-preservation-equals-reflection*:
fixes *Rel* :: $('a \times 'a)$ *set*
and *Pred* :: $'a \Rightarrow bool$
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-relation-impl-preservation-equals-reflection-of-binary-predicates*:
fixes *Rel* :: $('a \times 'a)$ *set*
and *Pred* :: $'a \Rightarrow 'b \Rightarrow bool$
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* :: ($'a \times 'a$) *set*
and *Pred* :: $'a \Rightarrow$ *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*: $\bigwedge$*Rel*. *rel-preserves-pred Rel Pred* $\Longrightarrow$ *rel-preserves-pred* (Rel^+) *Pred*
proof *clarify*
fix *Rel a b*
assume (a, b) $\in Rel^+$ **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* :: ($'a \times 'a$) *set*
and *Pred* :: $'a \Rightarrow 'b \Rightarrow$ *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*: $\bigwedge$*Rel*. *rel-preserves-binary-pred Rel Pred*
$\Longrightarrow$ *rel-preserves-binary-pred* (Rel^+) *Pred*
proof *clarify*
fix *Rel a b x*
assume (a, b) $\in Rel^+$ **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* :: ($'a \times 'a$) *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: ⋀Rel. rel-reflects-pred Rel Pred ⟹ 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: ⋀Rel. rel-reflects-binary-pred Rel Pred ⟹ 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 respection-and-closures:
  fixes Rel  :: ('a × 'a) set
    and Pred :: 'a ⇒ bool
  assumes respection: 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 respection 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*. *rel-respects-pred Rel Pred* $\Longrightarrow$ *rel-respects-pred* (*symcl Rel*) *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
qed
from *respection B*[**where** *Rel*=*Rel*] **show** *rel-respects-pred* (*symcl Rel*) *Pred*
by *blast*
have *C*: $\bigwedge$*Rel*. *rel-respects-pred Rel Pred* $\Longrightarrow$ *rel-respects-pred* (Rel^+) *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*. *rel-respects-binary-pred Rel Pred* $\Longrightarrow$ *rel-respects-binary-pred* (*symcl Rel*) *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*) ∈ *symcl Rel*
hence (*a*, *b*) ∈ *Rel* ∨ (*b*, *a*) ∈ *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*) ∈ *symcl Rel*
hence (*a*, *b*) ∈ *Rel* ∨ (*b*, *a*) ∈ *Rel*
by (*simp add*: *symcl-def*)
moreover assume *Pred b x*
ultimately show *Pred a x*
using *B2*
by *blast*
qed
qed
from *respection B*[**where** *Rel*=*Rel*] **show** *rel-respects-binary-pred* (*symcl Rel*) *Pred*
by *blast*
have *C*: $\bigwedge$*Rel*. *rel-respects-binary-pred Rel Pred* $\Longrightarrow$ *rel-respects-binary-pred* (Rel^{+}) *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
$P \longmapsto Cal\ Q \equiv Reductions\ Cal\ P\ Q$

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

primrec *nSteps*
:: *'proc* ⇒ *'proc processCalculus* ⇒ *nat* ⇒ *'proc* ⇒ *bool*
(‹- ⟼-ˆ- -› [*70*, *70*, *70*, *70*] *80*)
where
$P \longmapsto Cal^{0}\ Q \quad = (P = Q)\ |$
$P \longmapsto Cal^{Suc\ n}\ Q = (\exists P'.\ P \longmapsto Cal^{n}\ P' \wedge P' \longmapsto Cal\ Q)$

definition *steps*
:: *'proc* ⇒ *'proc processCalculus* ⇒ *'proc* ⇒ *bool*
(‹- ⟼-∗ -› [*70*, *70*, *70*] *80*)
where
$P \longmapsto Cal{*}\ Q \equiv \exists n.\ P \longmapsto Cal^{n}\ Q$

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

definition *divergent*
:: *'proc* ⇒ *'proc processCalculus* ⇒ *bool*
(‹- ⟼-ω› [*70*, *70*] *80*)
where
$P \longmapsto (Cal)\omega \equiv \forall P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists P''.\ P' \longmapsto Cal\ P'')$

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

lemma *steps-refl*:
fixes *Cal* :: *'proc processCalculus*
and *P* :: *'proc*
shows $P \longmapsto Cal{*}\ P$
proof −
have $P \longmapsto Cal^{0}\ P$
by *simp*
hence $\exists n.\ P \longmapsto Cal^{n}\ P$

by *blast*
thus $P \longmapsto Cal* \ P$
by (*simp add*: *steps-def*)
qed

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

lemma *step-to-steps*:
fixes Cal :: *'proc processCalculus*
and $P\ P'$ :: *'proc*
assumes *step*: $P \longmapsto Cal\ P'$
shows $P \longmapsto Cal*\ P'$
proof −
from *step* **have** $P \longmapsto Cal^{1}\ P'$
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 Cal :: *'proc processCalculus*
and *n1 n2* :: *nat*
shows $\forall P\ Q\ R.\ P \longmapsto Cal^{n1}\ Q \wedge Q \longmapsto Cal^{n2}\ R \longrightarrow P \longmapsto Cal^{(n1\ +\ n2)}\ R$
proof (*induct n2*, *simp*)
case (*Suc n*)
assume *IH*: $\forall P\ Q\ R.\ P \longmapsto Cal^{n1}\ Q \wedge Q \longmapsto Cal^{n}\ R \longrightarrow P \longmapsto Cal^{(n1\ +\ n)}\ R$
show *?case*
proof *clarify*
fix $P\ Q\ R$
assume $Q \longmapsto Cal^{Suc\ n}\ R$
from *this* **obtain** Q' **where** *A1*: $Q \longmapsto Cal^{n}\ Q'$ **and** *A2*: $Q' \longmapsto Cal\ R$
by *auto*
assume $P \longmapsto Cal^{n1}\ Q$
with *A1 IH* **have** $P \longmapsto Cal^{(n1\ +\ n)}\ Q'$
by *blast*
with *A2* **show** $P \longmapsto Cal^{(n1\ +\ Suc\ n)}\ R$
by *auto*
qed
qed

lemma *steps-add*:
fixes Cal :: *'proc processCalculus*
and $P\ Q\ R$ :: *'proc*
assumes *A1*: $P \longmapsto Cal*\ Q$
and *A2*: $Q \longmapsto Cal*\ R$
shows $P \longmapsto Cal*\ R$
proof −
from *A1* **obtain** *n1* **where** $P \longmapsto Cal^{n1}\ Q$
by (*auto simp add*: *steps-def*)
moreover from *A2* **obtain** *n2* **where** $Q \longmapsto Cal^{n2}\ R$
by (*auto simp add*: *steps-def*)
ultimately have $P \longmapsto Cal^{(n1\ +\ n2)}\ R$
using *nSteps-add*[**where** *Cal*=*Cal*]
by *blast*
thus $P \longmapsto Cal*\ R$
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 ⇒ 'barbs ⇒ bool (‹-↓-› [70, 70] 80)
```

```
abbreviation hasBarb
  :: 'proc ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs ⇒ bool
    (‹-↓<->-› [70, 70, 70] 80)
  where
  P↓<CWB>a ≡ HasBarb CWB P a
```

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

```
abbreviation reachesBarb
  :: 'proc ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs ⇒ bool
    (‹-⇓<->-› [70, 70, 70] 80)
  where
  P⇓<CWB>a ≡ ∃ P'. P ⟼(Calculus CWB)∗ P' ∧ P'↓<CWB>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 × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs set ⇒ bool
  where
  rel-preserves-barb-set Rel CWB Barbs ≡
   rel-preserves-binary-pred Rel (λP a. a ∈ Barbs ∧ P↓<CWB>a)
```

```
abbreviation rel-preserves-barbs
    :: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
  where
  rel-preserves-barbs Rel CWB ≡ rel-preserves-binary-pred Rel (HasBarb CWB)
```

```
lemma preservation-of-barbs-and-set-of-barbs:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  shows rel-preserves-barbs Rel CWB = (∀ 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 × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs set ⇒ bool
  where
  rel-reflects-barb-set Rel CWB Barbs ≡
   rel-reflects-binary-pred Rel (λP a. a ∈ Barbs ∧ P↓<CWB>a)
```

```
abbreviation rel-reflects-barbs
    :: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
  where
  rel-reflects-barbs Rel CWB ≡ rel-reflects-binary-pred Rel (HasBarb CWB)
```

```
lemma reflection-of-barbs-and-set-of-barbs:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  shows rel-reflects-barbs Rel CWB = (∀ 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 × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs set ⇒ bool
where
rel-respects-barb-set Rel CWB Barbs ≡
rel-preserves-barb-set Rel CWB Barbs ∧ *rel-reflects-barb-set Rel CWB Barbs*

abbreviation *rel-respects-barbs*
:: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
where
rel-respects-barbs Rel CWB ≡ *rel-preserves-barbs Rel CWB* ∧ *rel-reflects-barbs Rel CWB*

lemma *respection-of-barbs-and-set-of-barbs*:
fixes *Rel* :: ('proc × 'proc) set
and *CWB* :: ('proc, 'barbs) calculusWithBarbs
shows *rel-respects-barbs Rel CWB* = (∀ *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 × '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 × '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 × '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** $\mathit{rel\text{-}respects\text{-}barbs}\ (\mathit{Rel}^{+})\ \mathit{CWB}$
  using $\mathit{respection\text{-}of\text{-}binary\text{-}predicates\text{-}and\text{-}closures}(3)$[**where** $\mathit{Rel}=\mathit{Rel}$ **and** $\mathit{Pred}=\mathit{HasBarb}\ \mathit{CWB}$]
 by *blast*
next
from *respection* **show** $\mathit{rel\text{-}respects\text{-}barbs}\ (\mathit{symcl}\ (\mathit{Rel}^{=}))\ \mathit{CWB}$
  using $\mathit{respection\text{-}of\text{-}binary\text{-}predicates\text{-}and\text{-}closures}(4)$[**where** $\mathit{Rel}=\mathit{Rel}$ **and** $\mathit{Pred}=\mathit{HasBarb}\ \mathit{CWB}$]
 by *blast*
next
from *respection* **show** $\mathit{rel\text{-}respects\text{-}barbs}\ (\mathit{Rel}^{*})\ \mathit{CWB}$
  using $\mathit{respection\text{-}of\text{-}binary\text{-}predicates\text{-}and\text{-}closures}(5)$[**where** $\mathit{Rel}=\mathit{Rel}$ **and** $\mathit{Pred}=\mathit{HasBarb}\ \mathit{CWB}$]
 by *blast*
next
from *respection* **show** $\mathit{rel\text{-}respects\text{-}barbs}\ ((\mathit{symcl}\ (\mathit{Rel}^{=}))^{+})\ \mathit{CWB}$
  using $\mathit{respection\text{-}of\text{-}binary\text{-}predicates\text{-}and\text{-}closures}(6)$[**where** $\mathit{Rel}=\mathit{Rel}$ **and** $\mathit{Pred}=\mathit{HasBarb}\ \mathit{CWB}$]
 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
$\mathit{rel\text{-}weakly\text{-}preserves\text{-}barb\text{-}set}\ \mathit{Rel}\ \mathit{CWB}\ \mathit{Barbs} \equiv$
 $\mathit{rel\text{-}preserves\text{-}binary\text{-}pred}\ \mathit{Rel}\ (\lambda P\ a.\ a \in \mathit{Barbs} \wedge P{\Downarrow}{<}\mathit{CWB}{>}a)$

abbreviation *rel-weakly-preserves-barbs*
 $:: ('proc \times 'proc)\ set \Rightarrow ('proc, 'barbs)\ calculusWithBarbs \Rightarrow bool$
where
$\mathit{rel\text{-}weakly\text{-}preserves\text{-}barbs}\ \mathit{Rel}\ \mathit{CWB} \equiv \mathit{rel\text{-}preserves\text{-}binary\text{-}pred}\ \mathit{Rel}\ (\lambda P\ a.\ P{\Downarrow}{<}\mathit{CWB}{>}a)$

lemma *weak-preservation-of-barbs-and-set-of-barbs*:
fixes $\mathit{Rel} :: ('proc \times 'proc)\ set$
 and $\mathit{CWB} :: ('proc, 'barbs)\ calculusWithBarbs$
shows $\mathit{rel\text{-}weakly\text{-}preserves\text{-}barbs}\ \mathit{Rel}\ \mathit{CWB}$
  $= (\forall\, \mathit{Barbs}.\ \mathit{rel\text{-}weakly\text{-}preserves\text{-}barb\text{-}set}\ \mathit{Rel}\ \mathit{CWB}\ \mathit{Barbs})$
 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
$\mathit{rel\text{-}weakly\text{-}reflects\text{-}barb\text{-}set}\ \mathit{Rel}\ \mathit{CWB}\ \mathit{Barbs} \equiv$
 $\mathit{rel\text{-}reflects\text{-}binary\text{-}pred}\ \mathit{Rel}\ (\lambda P\ a.\ a \in \mathit{Barbs} \wedge P{\Downarrow}{<}\mathit{CWB}{>}a)$

abbreviation *rel-weakly-reflects-barbs*
 $:: ('proc \times 'proc)\ set \Rightarrow ('proc, 'barbs)\ calculusWithBarbs \Rightarrow bool$
where
$\mathit{rel\text{-}weakly\text{-}reflects\text{-}barbs}\ \mathit{Rel}\ \mathit{CWB} \equiv \mathit{rel\text{-}reflects\text{-}binary\text{-}pred}\ \mathit{Rel}\ (\lambda P\ a.\ P{\Downarrow}{<}\mathit{CWB}{>}a)$

lemma *weak-reflection-of-barbs-and-set-of-barbs*:
fixes $\mathit{Rel} :: ('proc \times 'proc)\ set$
 and $\mathit{CWB} :: ('proc, 'barbs)\ calculusWithBarbs$
shows $\mathit{rel\text{-}weakly\text{-}reflects\text{-}barbs}\ \mathit{Rel}\ \mathit{CWB} = (\forall\, \mathit{Barbs}.\ \mathit{rel\text{-}weakly\text{-}reflects\text{-}barb\text{-}set}\ \mathit{Rel}\ \mathit{CWB}\ \mathit{Barbs})$
 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* $(Rel^{=})$ *CWB*
and *rel-weakly-preserves-barbs* (Rel^{+}) *CWB*
and *rel-weakly-preserves-barbs* (Rel^{*}) *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* $(Rel^{=})$ *CWB*
and *rel-weakly-reflects-barbs* (Rel^{+}) *CWB*
and *rel-weakly-reflects-barbs* (Rel^{*}) *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* $(Rel^{=})$ *CWB*
and *rel-weakly-respects-barbs* (*symcl Rel*) *CWB*
and *rel-weakly-respects-barbs* (Rel^{+}) *CWB*
and *rel-weakly-respects-barbs* (*symcl* $(Rel^{=})$) *CWB*
and *rel-weakly-respects-barbs* (Rel^{*}) *CWB*
and *rel-weakly-respects-barbs* $((symcl\ (Rel^{=}))^{+})$ *CWB*
proof −
from *respection* **show** *rel-weakly-respects-barbs* $(Rel^{=})$ *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=λP a. P⇓<CWB>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=λP a. P⇓<CWB>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=λP a. P⇓<CWB>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=λP a. P⇓<CWB>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 × 'proc) set ⇒ 'proc processCalculus ⇒ bool
  where
  weak-reduction-simulation Rel Cal ≡
  ∀ P Q P'. (P, Q) ∈ Rel ∧ P ⟼Cal* P' ⟶ (∃ Q'. Q ⟼Cal* Q' ∧ (P', Q') ∈ Rel)
```

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

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

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

```
lemma weak-reduction-simulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
    and Cal :: 'proc processCalculus
  assumes simulation: weak-reduction-simulation Rel Cal
  shows weak-reduction-simulation (Rel^=) Cal
    and weak-reduction-simulation (Rel^+) Cal
    and weak-reduction-simulation (Rel^*) 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* $\Longrightarrow$ *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** $P \longmapsto Cal* \ P'$
thus $\exists Q'. \ Q \longmapsto Cal* \ Q' \wedge (P', Q') \in Rel^{+}$
proof (*induct arbitrary*: *P′*)
fix *Q P′*
assume $(P, Q) \in Rel$ **and** $P \longmapsto Cal* \ P'$
with *B1* **obtain** Q' **where** $Q \longmapsto Cal* \ Q'$ **and** $(P', Q') \in Rel$
by *blast*
thus $\exists Q'. \ Q \longmapsto Cal* \ Q' \wedge (P', Q') \in Rel^{+}$
by *auto*
next
case (*step Q R P′*)
assume $\bigwedge P'. \ P \longmapsto Cal* \ P' \Longrightarrow (\exists Q'. \ Q \longmapsto Cal* \ Q' \wedge (P', Q') \in Rel^{+})$
and $P \longmapsto Cal* \ P'$
from *this* **obtain** Q' **where** *B2*: $Q \longmapsto Cal* \ Q'$ **and** *B3*: $(P', Q') \in Rel^{+}$
by *blast*
assume $(Q, R) \in Rel$
with *B1 B2* **obtain** R' **where** *B4*: $R \longmapsto Cal* \ R'$ **and** *B5*: $(Q', R') \in Rel^{+}$
by *blast*
from *B3 B5* **have** $(P', R') \in Rel^{+}$
by *simp*
from *B4 this* **show** $\exists R'. \ R \longmapsto Cal* \ R' \wedge (P', R') \in Rel^{+}$
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* × *′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 × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
  where
  weak-barbed-preservation-cond Rel CWB ≡ ∀ P Q a. (P, Q) ∈ Rel ∧ P↓<CWB>a ⟶ Q⇓<CWB>a

lemma weak-preservation-of-barbs:
  fixes Rel :: ('proc × '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 ⟼(Calculus CWB)∗ P
    by (simp add: steps-refl)
  moreover assume P↓<CWB>a
  ultimately have P⇓<CWB>a
    by blast
  moreover assume (P, Q) ∈ Rel
  ultimately show Q⇓<CWB>a
    using preservation
    by blast
qed

lemma simulation-impl-equality-of-preservation-of-barbs-conditions:
  fixes Rel :: ('proc × 'proc) set
    and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: weak-reduction-simulation Rel (Calculus CWB)
  shows rel-weakly-preserves-barbs Rel CWB = weak-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) ∈ Rel and P ⟼(Calculus CWB)∗ P'
    with simulation obtain Q' where A1: Q ⟼(Calculus CWB)∗ Q' and A2: (P', Q') ∈ Rel
      by blast
    assume P'↓<CWB>a
    with A2 condition obtain Q'' where A3: Q' ⟼(Calculus CWB)∗ Q'' and A4: Q''↓<CWB>a
      by blast
    from A1 A3 have Q ⟼(Calculus CWB)∗ Q''
      by (rule steps-add)
    with A4 show Q⇓<CWB>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 × 'proc) set ⇒ 'proc processCalculus ⇒ bool
  where
  strong-reduction-simulation Rel Cal ≡
```

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

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-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 $P \longmapsto Cal{*}\ P'$
from *this* **obtain** n **where** $P \longmapsto Cal^n\ P'$
by (*auto simp add*: *steps-def*)
thus $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel$
proof (*induct n arbitrary*: P')
case *0*
assume $P \longmapsto Cal^0\ P'$
hence $P = P'$
by (*simp add*: *steps-refl*)
moreover have $Q \longmapsto Cal{*}\ Q$
by (*rule steps-refl*)
ultimately show $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel$
using *A1*
by *blast*
next
case (*Suc n* P'')
assume $P \longmapsto Cal^{Suc\ n}\ P''$
from *this* **obtain** P' **where** *A2*: $P \longmapsto Cal^n P'$ **and** *A3*: $P' \longmapsto Cal\ P''$
by *auto*
assume $\bigwedge P'.\ P \longmapsto Cal^n\ P' \Longrightarrow \exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel$
with *A2* **obtain** Q' **where** *A4*: $Q \longmapsto Cal{*}\ Q'$ **and** *A5*: $(P',\ Q') \in Rel$
by *blast*
from *simulation A5 A3* **obtain** Q'' **where** *A6*: $Q' \longmapsto Cal\ Q''$ **and** *A7*: $(P'',\ Q'') \in Rel$
by *blast*
from *A4 A6* **have** $Q \longmapsto Cal{*}\ Q''$
using *steps-add*[**where** $P{=}Q$ **and** $Q{=}Q'$ **and** $R{=}Q''$]
by (*simp add*: *step-to-steps*)
with *A7* **show** $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P'',\ Q') \in Rel$
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** $P \longmapsto (Calculus\ CWB){*}\ P'$
with *simulation* **obtain** Q' **where** *A1*: $Q \longmapsto (Calculus\ CWB){*}\ Q'$ **and** *A2*: $(P',\ Q') \in Rel$
using *strong-impl-weak-reduction-simulation*[**where** *Rel*=*Rel* **and** *Cal*=*Calculus CWB*]

by *blast*
assume $P' \downarrow {<} CWB {>} a$
with *simulation A2* **have** $Q' \downarrow {<} CWB {>} a$
by *blast*
with *A1* **show** $Q \Downarrow {<} CWB {>} a$
by *blast*
qed

lemma *strong-impl-weak-barbed-simulation*:
fixes *Rel* :: $('proc \times 'proc)$ *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* :: $('proc \times 'proc)$ *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*
$\Longrightarrow$ *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** $P \longmapsto Cal\ P'$
thus $\exists Q'. Q \longmapsto Cal\ Q' \wedge (P', Q') \in Rel^+$
proof (*induct arbitrary*: P')
fix $Q\ P'$
assume $(P, Q) \in Rel$ **and** $P \longmapsto Cal\ P'$
with *B1* **obtain** Q' **where** $Q \longmapsto Cal\ Q'$ **and** $(P', Q') \in Rel$
by *blast*
thus $\exists Q'. Q \longmapsto Cal\ Q' \wedge (P', Q') \in Rel^+$
by *auto*
next
case (*step* $Q\ R\ P'$)
assume $\bigwedge P'. P \longmapsto Cal\ P' \Longrightarrow (\exists Q'. Q \longmapsto Cal\ Q' \wedge (P', Q') \in Rel^+)$
and $P \longmapsto Cal\ P'$
from *this* **obtain** Q' **where** *B2*: $Q \longmapsto Cal\ Q'$ **and** *B3*: $(P', Q') \in Rel^+$
by *blast*
assume $(Q, R) \in Rel$
with *B1 B2* **obtain** R' **where** *B4*: $R \longmapsto Cal\ R'$ **and** *B5*: $(Q', R') \in Rel^+$
by *blast*
from *B3 B5* **have** $(P', R') \in Rel^+$
by *simp*
with *B4* **show** $\exists R'. R \longmapsto Cal\ R' \wedge (P', R') \in Rel^+$
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* :: $('proc \times 'proc)$ *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 Contrasimulation

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

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

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

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

lemma *weak-reduction-contrasimulation-and-closures*:
  fixes *Rel* :: $('proc \times 'proc)$ *set*
  and *Cal* :: *'proc processCalculus*
  assumes *contrasimulation*: *weak-reduction-contrasimulation Rel Cal*
  shows *weak-reduction-contrasimulation* $(Rel^=)$ *Cal*
  and *weak-reduction-contrasimulation* (Rel^+) *Cal*
  and *weak-reduction-contrasimulation* (Rel^*) *Cal*
proof −
  from *contrasimulation* **show** *A*: *weak-reduction-contrasimulation* $(Rel^=)$ *Cal*
  by (*auto simp add*: *refl*, *blast*)
  have *B*: $\bigwedge$*Rel*. *weak-reduction-contrasimulation Rel Cal*
    $\Longrightarrow$ *weak-reduction-contrasimulation* (Rel^+) *Cal*

proof *clarify*
fix *Rel P Q P′*
assume *B1*: *weak-reduction-contrasimulation Rel Cal*
assume $(P, Q) \in Rel^+$ **and** $P \longmapsto Cal* \; P'$
thus $\exists Q'. \; Q \longmapsto Cal* \; Q' \wedge (Q', P') \in Rel^+$
proof (*induct arbitrary*: *P′*)
fix *Q P′*
assume $(P, Q) \in Rel$ **and** $P \longmapsto Cal* \; P'$
with *B1* **obtain** Q' **where** $Q \longmapsto Cal* \; Q'$ **and** $(Q', P') \in Rel$
by *blast*
thus $\exists Q'. \; Q \longmapsto Cal* \; Q' \wedge (Q', P') \in Rel^+$
by *auto*
next
case (*step Q R P′*)
assume $\bigwedge P'. \; P \longmapsto Cal* \; P' \Longrightarrow (\exists Q'. \; Q \longmapsto Cal* \; Q' \wedge (Q', P') \in Rel^+)$
and $P \longmapsto Cal* \; P'$
from *this* **obtain** Q' **where** *B2*: $Q \longmapsto Cal* \; Q'$ **and** *B3*: $(Q', P') \in Rel^+$
by *blast*
assume $(Q, R) \in Rel$
with *B1 B2* **obtain** R' **where** *B4*: $R \longmapsto Cal* \; R'$ **and** *B5*: $(R', Q') \in Rel^+$
by *blast*
from *B5 B3* **have** $(R', P') \in Rel^+$
by *simp*
with *B4* **show** $\exists R'. \; R \longmapsto Cal* \; R' \wedge (R', P') \in Rel^+$
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 exits some Q' such that Q evolves to Q' and (Q', P') in R.

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

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

abbreviation *weak-barbed-coupled-simulation*
  :: $('proc \times 'proc)$ *set* $\Rightarrow$ $('proc,\ 'barbs)$ *calculusWithBarbs* $\Rightarrow$ *bool*
 where
 weak-barbed-coupled-simulation Rel CWB $\equiv$
 weak-reduction-coupled-simulation Rel (Calculus CWB) $\wedge$ *rel-weakly-preserves-barbs Rel 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 processCalculus*
 shows *weak-reduction-coupled-simulation Rel Cal*
   = (*weak-reduction-simulation Rel Cal* $\wedge$ *weak-reduction-contrasimulation Rel Cal*)
  by *blast*

lemma *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation*:
 fixes *Rel* :: $('proc \times 'proc)$ *set*
  and *CWB* :: $('proc,\ 'barbs)$ *calculusWithBarbs*
 shows *weak-barbed-coupled-simulation Rel CWB*
   = (*weak-barbed-simulation Rel CWB* $\wedge$ *weak-barbed-contrasimulation Rel 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 processCalculus*
 assumes *coupledSimulation*: *weak-reduction-coupled-simulation Rel Cal*
 shows *weak-reduction-coupled-simulation* $(Rel^{=})$ *Cal*
  and *weak-reduction-coupled-simulation* (Rel^{+}) *Cal*
  and *weak-reduction-coupled-simulation* (Rel^{*}) *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* :: $('proc,\ 'barbs)$ *calculusWithBarbs*
 assumes *coupledSimulation*: *weak-barbed-coupled-simulation Rel CWB*
 shows *weak-barbed-coupled-simulation* $(Rel^{=})$ *CWB*
  and *weak-barbed-coupled-simulation* (Rel^{+}) *CWB*
  and *weak-barbed-coupled-simulation* (Rel^{*}) *CWB*
proof –
 from *coupledSimulation* **show** *weak-barbed-coupled-simulation* $(Rel^{=})$ *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 P" and 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 $\equiv$
$(\forall P\ Q\ P'.\ (P,\ Q) \in Rel \wedge P \longmapsto Cal{*}\ P' \longrightarrow (\exists Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel))$
$\wedge (\forall P\ Q\ Q'.\ (P,\ Q) \in Rel \wedge Q \longmapsto Cal{*}\ Q'$
$\longrightarrow (\exists P''\ Q''.\ P \longmapsto Cal{*}\ P'' \wedge Q' \longmapsto Cal{*}\ Q'' \wedge (P'',\ Q'') \in Rel))$

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

```
  proof clarify
    fix P Q
    have Q ⟼Cal∗ Q
      by (rule steps-refl)
    moreover assume (P, Q) ∈ Rel
    ultimately show (P, Q) ∈ cSim-cs Rel Cal ∧ (Q, P) ∈ 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 × '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 ∀ P Q. (P, Q) ∈ Rel ⟶ (P, Q) ∈ cSim-cs Rel (Calculus CWB)
                              ∧ (Q, P) ∈ 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) ∈ cSim-cs Rel (Calculus CWB) and P ⟼(Calculus CWB)∗ P' and P'↓<CWB>a
      thus Q⇓<CWB>a
      proof (induct arbitrary: P')
        case (left Q Q' P P')
        assume (P, Q') ∈ Rel and P ⟼(Calculus CWB)∗ P' and P'↓<CWB>a
        with corrSim obtain Q'' where A1: Q' ⟼(Calculus CWB)∗ Q'' and A2: Q''↓<CWB>a
          by blast
        assume Q ⟼(Calculus CWB)∗ Q'
        from this A1 have Q ⟼(Calculus CWB)∗ Q''
          by (rule steps-add)
        with A2 show Q⇓<CWB>a
          by blast
      next
        case (right P P' Q P'')
        assume (Q, P) ∈ Rel
        moreover assume P ⟼(Calculus CWB)∗ P' and P' ⟼(Calculus CWB)∗ P''
        hence P ⟼(Calculus CWB)∗ P''
          by (rule steps-add)
        moreover assume P''↓<CWB>a
        ultimately show Q⇓<CWB>a
            using corrSim
          by blast
      next
        case (trans P Q R P')
        assume ⋀P'. P ⟼(Calculus CWB)∗ P' ⟹ P'↓<CWB>a ⟹ Q⇓<CWB>a
           and P ⟼(Calculus CWB)∗ P' and P'↓<CWB>a
           and ⋀Q'. Q ⟼(Calculus CWB)∗ Q' ⟹ Q'↓<CWB>a ⟹ R⇓<CWB>a
        thus R⇓<CWB>a
          by blast
      qed
    qed
  qed
```

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

lemma *reduction-correspondence-simulation-condition-trans*:
  fixes *Cal* :: *'proc processCalculus*
   and *P Q R* :: *'proc*
   and *Rel* :: (*'proc* × *'proc*) *set*
  assumes *A1*: $\forall Q'.\ Q \longmapsto Cal{*}\ Q' \longrightarrow (\exists P''\ Q''.\ P \longmapsto Cal{*}\ P'' \wedge Q' \longmapsto Cal{*}\ Q'' \wedge (P'',\ Q'') \in Rel)$
    and *A2*: $\forall R'.\ R \longmapsto Cal{*}\ R' \longrightarrow (\exists Q''\ R''.\ Q \longmapsto Cal{*}\ Q'' \wedge R' \longmapsto Cal{*}\ R'' \wedge (Q'',\ R'') \in Rel)$
    and *A3*: *weak-reduction-simulation Rel Cal*
    and *A4*: *trans Rel*
  shows $\forall R'.\ R \longmapsto Cal{*}\ R' \longrightarrow (\exists P''\ R''.\ P \longmapsto Cal{*}\ P'' \wedge R' \longmapsto Cal{*}\ R'' \wedge (P'',\ R'') \in Rel)$
proof *clarify*
  fix R'
  assume $R \longmapsto Cal{*}\ R'$
  with *A2* **obtain** $Q''\ R''$ **where** *A5*: $Q \longmapsto Cal{*}\ Q''$ **and** *A6*: $R' \longmapsto Cal{*}\ R''$
      and *A7*: $(Q'',\ R'') \in Rel$
   by *blast*
  from *A1 A5* **obtain** $P'''\ Q'''$ **where** *A8*: $P \longmapsto Cal{*}\ P'''$ **and** *A9*: $Q'' \longmapsto Cal{*}\ Q'''$
      and *A10*: $(P''',\ Q''') \in Rel$
   by *blast*
  from *A3 A7 A9* **obtain** R''' **where** *A11*: $R'' \longmapsto Cal{*}\ R'''$ **and** *A12*: $(Q''',\ R''') \in Rel$
   by *blast*
  from *A6 A11* **have** *A13*: $R' \longmapsto Cal{*}\ R'''$
   by (*rule steps-add*[**where** $P=R'$ **and** $Q=R''$ **and** $R=R'''$])
  from *A4 A10 A12* **have** $(P''',\ R''') \in Rel$
    unfolding *trans-def*
   by *blast*
  with *A8 A13* **show** $\exists P''\ R''.\ P \longmapsto Cal{*}\ P'' \wedge R' \longmapsto Cal{*}\ R'' \wedge (P'',\ R'') \in Rel$
   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 *Rel* :: (*'proc* × *'proc*) *set*
   and *Cal* :: *'proc processCalculus*
  assumes *corrSim*: *weak-reduction-correspondence-simulation Rel Cal*
  shows *weak-reduction-correspondence-simulation* ($Rel^=$) *Cal*
   and *weak-reduction-correspondence-simulation* (Rel^+) *Cal*
   and *weak-reduction-correspondence-simulation* (Rel^*) *Cal*
proof –
  show *A*: *weak-reduction-correspondence-simulation* ($Rel^=$) *Cal*
  proof
   from *corrSim* **show** *weak-reduction-simulation* ($Rel^=$) *Cal*
     using *weak-reduction-simulation-and-closures*(*1*)[**where** *Rel*=*Rel* **and** *Cal*=*Cal*]
    by *blast*
  next
   show $\forall P\ Q\ Q'.\ (P,\ Q) \in Rel^= \wedge Q \longmapsto Cal{*}\ Q'$
     $\longrightarrow (\exists P''\ Q''.\ P \longmapsto Cal{*}\ P'' \wedge Q' \longmapsto Cal{*}\ Q'' \wedge (P'',\ Q'') \in Rel^=)$
   proof *clarify*
    fix $P\ Q\ Q'$
    assume $(P,\ Q) \in Rel^=$ **and** *A1*: $Q \longmapsto Cal{*}\ Q'$
    moreover have $P = Q \Longrightarrow \exists P''\ Q''.\ P \longmapsto Cal{*}\ P'' \wedge Q' \longmapsto Cal{*}\ Q'' \wedge (P'',\ Q'') \in Rel^=$
    proof –

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

lemma *weak-barbed-correspondence-simulation-and-closures*:
fixes *Rel* :: $('proc \times 'proc)$ *set*
and *CWB* :: $('proc, 'barbs)$ *calculusWithBarbs*
assumes *corrSim*: *weak-barbed-correspondence-simulation Rel CWB*
shows *weak-barbed-correspondence-simulation* $(Rel^{=})$ *CWB*
and *weak-barbed-correspondence-simulation* (Rel^{+}) *CWB*
and *weak-barbed-correspondence-simulation* (Rel^{*}) *CWB*
proof −
from *corrSim* **show** *weak-barbed-correspondence-simulation* $(Rel^{=})$ *CWB*
using *weak-reduction-correspondence-simulation-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 *corrSim* **show** *weak-barbed-correspondence-simulation* (Rel^{+}) *CWB*
using *weak-reduction-correspondence-simulation-and-closures*(*2*)[**where** *Rel=Rel*
and *Cal=Calculus CWB*]
weak-respection-of-barbs-and-closures(*3*)[**where** *Rel=Rel* **and** *CWB=CWB*]
by *blast*
next
from *corrSim* **show** *weak-barbed-correspondence-simulation* (Rel^{*}) *CWB*
using *weak-reduction-correspondence-simulation-and-closures*(*3*)[**where** *Rel=Rel*
and *Cal=Calculus CWB*]
weak-respection-of-barbs-and-closures(*5*)[**where** *Rel=Rel* **and** *CWB=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*
:: $('proc \times 'proc)$ *set* $\Rightarrow$ *'proc processCalculus* $\Rightarrow$ *bool*
where
weak-reduction-bisimulation Rel Cal $\equiv$
$(\forall\, P\ Q\ P'.\ (P, Q) \in Rel \wedge P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P', Q') \in Rel))$
$\wedge\ (\forall\, P\ Q\ Q'.\ (P, Q) \in Rel \wedge Q \longmapsto Cal{*}\ Q' \longrightarrow (\exists\, P'.\ P \longmapsto Cal{*}\ P' \wedge (P', Q') \in Rel))$

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

abbreviation *weak-barbed-bisimulation*
:: $('proc \times 'proc)$ *set* $\Rightarrow$ $('proc, 'barbs)$ *calculusWithBarbs* $\Rightarrow$ *bool*
where
weak-barbed-bisimulation Rel CWB $\equiv$

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

A symetric 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^{-1}) *Cal*
  shows *weak-reduction-bisimulation Rel Cal*
proof *auto*
  fix $P\ Q\ P'$
  assume $(P, Q) \in Rel$ **and** $P \longmapsto Cal{*}\ P'$
  with *sim* **show** $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P', Q') \in Rel$
    by *simp*
next
  fix $P\ Q\ Q'$
  assume $(P, Q) \in Rel$
  hence $(Q, P) \in Rel^{-1}$
    by *simp*
  moreover assume $Q \longmapsto Cal{*}\ Q'$
  ultimately obtain P' **where** *A1*: $P \longmapsto Cal{*}\ P'$ **and** *A2*: $(Q', P') \in Rel^{-1}$
      using *simInv*
    by *blast*
  from *A2* **have** $(P', Q') \in Rel$
    by *induct*
  with *A1* **show** $\exists\, P'.\ P \longmapsto Cal{*}\ P' \wedge (P', Q') \in Rel$
    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^{-1}) *Cal*
proof *clarify*
  fix $P\ Q\ P'$
  assume $(Q, P) \in Rel$
  moreover assume $P \longmapsto Cal{*}\ P'$
  ultimately obtain Q' **where** *A1*: $Q \longmapsto Cal{*}\ Q'$ **and** *A2*: $(Q', P') \in Rel$
      using *bisim*
    by *blast*

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

lemma *weak-reduction-simulations-iff-bisimulation*:
fixes *Rel* :: ('*proc* × '*proc*) *set*
and *Cal* :: '*proc processCalculus*
shows (*weak-reduction-simulation Rel Cal* ∧ *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* × '*proc*) *set*
and *CWB* :: ('*proc*, '*barbs*) *calculusWithBarbs*
shows (*weak-barbed-simulation Rel CWB* ∧ *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* ∧ *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* × '*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 $\forall\, P\ Q\ Q'.\ (P,\ Q) \in Rel \wedge Q \longmapsto Cal{*}\ Q'$
$\longrightarrow (\exists\, P''\ Q''.\ P \longmapsto Cal{*}\ P'' \wedge Q' \longmapsto Cal{*}\ Q'' \wedge (P'',\ Q'') \in Rel)$
proof *clarify*
fix *P Q Q'*

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

lemma *weak-barbed-bisimulation-is-correspondence-simulation*:
fixes *Rel* :: ('*proc* × '*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* × '*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*
$\Longrightarrow$ *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*
$\Longrightarrow$ *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 $\forall P\ Q\ Q'.\ (P, Q) \in Rel^{+} \wedge Q \longmapsto Cal{*}\ Q'$
$\longrightarrow (\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P', Q') \in Rel^{+})$
proof *clarify*
fix $P\ Q\ Q'$
assume $(P, Q) \in Rel^{+}$ **and** $Q \longmapsto Cal{*}\ Q'$
thus $\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P', Q') \in Rel^{+}$
proof (*induct arbitrary*: Q')
fix $Q\ Q'$
assume $(P, Q) \in Rel$ **and** $Q \longmapsto Cal{*}\ Q'$
with *C1* **obtain** P' **where** $P \longmapsto Cal{*}\ P'$ **and** $(P', Q') \in Rel$
by *blast*
thus $\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P', Q') \in Rel^{+}$

by *auto*
next
case (*step Q R R'*)
assume $(Q, R) \in Rel$ **and** $R \longmapsto Cal* \ R'$
with *C1* **obtain** Q' **where** *C2*: $Q \longmapsto Cal* \ Q'$ **and** *C3*: $(Q', R') \in Rel^+$
by *blast*
assume $\bigwedge Q'. \ Q \longmapsto Cal* \ Q' \Longrightarrow \exists P'. \ P \longmapsto Cal* \ P' \wedge (P', Q') \in Rel^+$
with *C2* **obtain** P' **where** *C4*: $P \longmapsto Cal* \ P'$ **and** *C5*: $(P', Q') \in Rel^+$
by *blast*
from *C5 C3* **have** $(P', R') \in Rel^+$
by *simp*
with *C4* **show** $\exists P'. \ P \longmapsto Cal* \ P' \wedge (P', R') \in Rel^+$
by *blast*
qed
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* :: $('proc \times 'proc)$ *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*
  $:: ('proc \times 'proc)\ set \Rightarrow 'proc\ processCalculus \Rightarrow bool$
 where
 strong-reduction-bisimulation Rel Cal ≡
 $(\forall P\ Q\ P'.\ (P,\ Q) \in Rel \wedge P \longmapsto Cal\ P' \longrightarrow (\exists Q'.\ Q \longmapsto Cal\ Q' \wedge (P',\ Q') \in Rel))$
 $\wedge (\forall P\ Q\ Q'.\ (P,\ Q) \in Rel \wedge Q \longmapsto Cal\ Q' \longrightarrow (\exists P'.\ P \longmapsto Cal\ P' \wedge (P',\ Q') \in Rel))$

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

abbreviation *strong-barbed-bisimulation*
  $:: ('proc \times 'proc)\ set \Rightarrow ('proc,\ 'barbs)\ calculusWithBarbs \Rightarrow bool$
 where
 strong-barbed-bisimulation Rel CWB ≡
  strong-reduction-bisimulation Rel (*Calculus CWB*) ∧ *rel-respects-barbs Rel CWB*

A symetric strong simulation is a strong bisimulation.

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

lemma *symm-strong-barbed-simulation-is-bisimulation*:
 fixes $Rel :: ('proc \times 'proc)\ set$
  and $CWB :: ('proc,\ 'barbs)\ calculusWithBarbs$
 assumes *sym Rel*
   and *strong-barbed-simulation Rel CWB*
 shows *strong-barbed-bisimulation Rel 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 $Cal :: 'proc\ processCalculus$
 assumes *sim*:   *strong-reduction-simulation Rel Cal*
   and *simInv*: *strong-reduction-simulation* (Rel^{-1}) *Cal*
 shows *strong-reduction-bisimulation Rel Cal*
proof *auto*
 fix $P\ Q\ P'$
 assume $(P,\ Q) \in Rel$ **and** $P \longmapsto Cal\ P'$
 with *sim* **show** $\exists Q'.\ Q \longmapsto Cal\ Q' \wedge (P',\ Q') \in Rel$
  by *simp*
next

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

lemma *strong-reduction-bisimulations-impl-inverse-is-simulation*:
fixes Rel :: $('proc \times 'proc)\ set$
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 $(Q,\ P) \in Rel$
moreover assume $P \longmapsto Cal\ P'$
ultimately obtain Q' **where** $A1$: $Q \longmapsto Cal\ Q'$ **and** $A2$: $(Q',\ P') \in Rel$
using *bisim*
by *blast*
from $A2$ **have** $(P',\ Q') \in Rel^{-1}$
by *simp*
with $A1$ **show** $\exists Q'.\ Q \longmapsto Cal\ Q' \wedge (P',\ Q') \in Rel^{-1}$
by *blast*
qed

lemma *strong-reduction-simulations-iff-bisimulation*:
fixes Rel :: $('proc \times 'proc)\ set$
and Cal :: $'proc\ processCalculus$
shows (*strong-reduction-simulation Rel Cal* $\wedge$ *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 :: $('proc \times 'proc)\ set$
and CWB :: $('proc,\ 'barbs)\ calculusWithBarbs$
shows (*strong-barbed-simulation Rel CWB* $\wedge$ *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* :: $('proc \times 'proc)$ *set*
and *Cal* :: *'proc 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 $\forall P\ Q\ Q'.\ (P,\ Q) \in Rel \wedge Q \longmapsto Cal{*}\ Q' \longrightarrow (\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P',\ Q') \in Rel)$
proof *clarify*
fix $P\ Q\ Q'$
assume *A1*: $(P,\ Q) \in Rel$
assume $Q \longmapsto Cal{*}\ Q'$
from *this* **obtain** n **where** $Q \longmapsto Cal^{n}\ Q'$
by (*auto simp add*: *steps-def*)
thus $\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P',\ Q') \in Rel$
proof (*induct n arbitrary*: Q')
case *0*
assume $Q \longmapsto Cal^{0}\ Q'$
hence $Q = Q'$
by (*simp add*: *steps-refl*)
moreover have $P \longmapsto Cal{*}\ P$
by (*rule steps-refl*)
ultimately show $\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P',\ Q') \in Rel$
using *A1*
by *blast*
next
case (*Suc n* Q'')
assume $Q \longmapsto Cal^{Suc\ n}\ Q''$
from *this* **obtain** Q' **where** *A2*: $Q \longmapsto Cal^{n} Q'$ **and** *A3*: $Q' \longmapsto Cal\ Q''$
by *auto*
assume $\bigwedge Q'.\ Q \longmapsto Cal^{n}\ Q' \Longrightarrow \exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P',\ Q') \in Rel$
with *A2* **obtain** P' **where** *A4*: $P \longmapsto Cal{*}\ P'$ **and** *A5*: $(P',\ Q') \in Rel$
by *blast*
from *bisim A5 A3* **obtain** P'' **where** *A6*: $P' \longmapsto Cal\ P''$ **and** *A7*: $(P'',\ Q'') \in Rel$
by *blast*
from *A4 A6* **have** $P \longmapsto Cal{*}\ P''$
using *steps-add*[**where** P=P **and** Q=P' **and** R=P'']
by (*simp add*: *step-to-steps*)
with *A7* **show** $\exists P'.\ P \longmapsto Cal{*}\ P' \wedge (P',\ Q'') \in Rel$
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*) ∈ *Rel* **and** *Q* ⟼(*Calculus CWB*)∗ *Q'*
with *bisim* **obtain** *P'* **where** *A1*: *P* ⟼(*Calculus CWB*)∗ *P'* **and** *A2*: (*P'*, *Q'*) ∈ *Rel*
using *strong-impl-weak-reduction-bisimulation*[**where** *Rel*=*Rel* **and** *Cal*=*Calculus CWB*]
by *blast*
assume *Q'*↓<*CWB*>*a*
with *bisim A2* **have** *P'*↓<*CWB*>*a*
by *blast*
with *A1* **show** *P*⇓<*CWB*>*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*: *strong-reduction-bisimulation* ($Rel^=$) *Cal*
by (*auto simp add*: *refl*, *blast*+)
have *B*: ⋀*Rel*. *strong-reduction-bisimulation Rel Cal*
⟹ *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*: ⋀*Rel*. *strong-reduction-bisimulation Rel Cal*
⟹ *strong-reduction-bisimulation* (Rel^+) *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 ∀ P Q Q'. (P, Q) ∈ Rel^+ ∧ Q ⟼Cal Q'
         ⟶ (∃ P'. P ⟼Cal P' ∧ (P', Q') ∈ Rel^+)
    proof clarify
      fix P Q Q'
      assume (P, Q) ∈ Rel^+ and Q ⟼Cal Q'
      thus ∃ P'. P ⟼Cal P' ∧ (P', Q') ∈ Rel^+
      proof (induct arbitrary: Q')
        fix Q Q'
        assume (P, Q) ∈ Rel and Q ⟼Cal Q'
        with C1 obtain P' where P ⟼Cal P' and (P', Q') ∈ Rel
          by blast
        thus ∃ P'. P ⟼Cal P' ∧ (P', Q') ∈ 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 ⋀Q'. Q ⟼Cal Q' ⟹ ∃ P'. P ⟼Cal P' ∧ (P', Q') ∈ 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 ∃ P'. P ⟼Cal P' ∧ (P', R') ∈ 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 × '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]
          respection-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*]
respection-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*]
respection-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*]
respection-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*]
respection-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*]
respection-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 ⟼Cal∗ P′
    thus P′ 𝓡↦<Rel,Cal> P
      by (rule stepsClosure.steps)
  next
    case (trans P Q R)
    assume P 𝓡↦<Rel,Cal> Q and Q 𝓡↦<Rel,Cal> R
    thus P 𝓡↦<Rel,Cal> R
      by (rule stepsClosure.trans)
  qed
next
  fix P Q
  assume P 𝓡↦<Rel,Cal> Q
  thus P 𝓡↦<stepsClosure Rel Cal,Cal> Q
    by (rule stepsClosure.rel)
qed
```

The steps closure is a preorder.

```
lemma stepsClosure-refl:
  fixes Rel :: (′a × ′a) set
    and Cal :: ′a processCalculus
  shows refl (stepsClosure Rel Cal)
    unfolding refl-on-def
proof auto
  fix P
  have P ⟼Cal∗ P
    by (rule steps-refl)
  thus P 𝓡↦<Rel,Cal> P
    by (rule stepsClosure.steps)
qed

lemma refl-trans-closure-of-rel-impl-steps-closure:
  fixes Rel :: (′a × ′a) set
    and Cal :: ′a processCalculus
    and P Q :: ′a
  assumes (P, Q) ∈ Rel∗
  shows P 𝓡↦<Rel,Cal> Q
    using assms
proof induct
  show P 𝓡↦<Rel,Cal> P
      using stepsClosure-refl[of Rel Cal]
      unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume (Q, R) ∈ Rel and P 𝓡↦<Rel,Cal> Q
  thus P 𝓡↦<Rel,Cal> 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 × ′a) set
    and Cal :: ′a processCalculus
  shows weak-reduction-simulation (stepsClosure Rel Cal) Cal
proof clarify
  fix P Q P′
  assume P 𝓡↦<Rel,Cal> Q and P ⟼Cal∗ P′
```

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

If Rel is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of Rel is a contrasimulation.

lemma *inverse-contrasimulation-impl-reverse-pair-in-steps-closure*:
fixes *Rel* :: $('a \times 'a)$ *set*
and *Cal* :: $'a$ *processCalculus*
and $P \ Q$ :: $'a$
assumes *con*: *weak-reduction-contrasimulation* (Rel^{-1}) *Cal*
and *pair*: $(P, Q) \in Rel$
shows $Q \ \mathcal{R}{\mapsto}{<}Rel,Cal{>} \ P$
proof −
from *pair* **have** $(Q, P) \in Rel^{-1}$
by *simp*
moreover have $Q \longmapsto Cal* \ Q$
by (*rule steps-refl*)
ultimately obtain P' **where** *A1*: $P \longmapsto Cal* \ P'$ **and** *A2*: $(P', Q) \in Rel^{-1}$
using *con*
by *blast*
from *A2* **have** $Q \ \mathcal{R}{\mapsto}{<}Rel,Cal{>} \ P'$
by (*simp add*: *stepsClosure.rel*)

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

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

Accordingly, if Rel is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of Rel is a coupled simulation.

lemma *simulation-and-inverse-contrasimulation-impl-steps-closure-is-coupled-simulation*:
fixes Rel :: $('a \times 'a)$ *set*
and Cal :: $'a$ *processCalculus*
assumes *sim*: *weak-reduction-simulation Rel Cal*
and *con*: *weak-reduction-contrasimulation* (Rel^{-1}) *Cal*
shows *weak-reduction-coupled-simulation* (*stepsClosure Rel Cal*) *Cal*
using *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 conlude 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*: $P\ \mathcal{R}{\mapsto}{<}Rel,Cal{>}\ Q$
and *A2*: *weak-reduction-simulation Rel Cal*
shows $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P,\ Q') \in Rel^*$
proof −
have $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
using *A1*
proof *induct*
case (*rel P Q*)
assume $(P,\ Q) \in Rel$
with *A2* **have** $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel)$
by *blast*
thus $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
by *blast*
next
case (*steps P P′*)
assume *A*: $P \longmapsto Cal{*}\ P'$
show $\forall\, P''.\ P' \longmapsto Cal{*}\ P'' \longrightarrow (\exists\, Q'.\ P \longmapsto Cal{*}\ Q' \wedge (P'',\ Q') \in Rel^*)$
proof *clarify*
fix P''
assume $P' \longmapsto Cal{*}\ P''$
with *A* **have** $P \longmapsto Cal{*}\ P''$
by (*rule steps-add*)
moreover have $(P'',\ P'') \in Rel^*$
by *simp*
ultimately show $\exists\, Q'.\ P \longmapsto Cal{*}\ Q' \wedge (P'',\ Q') \in Rel^*$
by *blast*
qed
next
case (*trans P Q R*)
assume *A1*: $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
and *A2*: $\forall\, Q'.\ Q \longmapsto Cal{*}\ Q' \longrightarrow (\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (Q',\ R') \in Rel^*)$
show $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (P',\ R') \in Rel^*)$
proof *clarify*
fix P'
assume $P \longmapsto Cal{*}\ P'$
with *A1* **obtain** Q' **where** *A3*: $Q \longmapsto Cal{*}\ Q'$ **and** *A4*: $(P',\ Q') \in Rel^*$
by *blast*
from *A2 A3* **obtain** R' **where** *A5*: $R \longmapsto Cal{*}\ R'$ **and** *A6*: $(Q',\ R') \in Rel^*$
by *blast*
from *A4 A6* **have** $(P',\ R') \in Rel^*$
by *simp*
with *A5* **show** $\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (P',\ R') \in Rel^*$
by *blast*
qed
qed
moreover have $P \longmapsto Cal{*}\ P$
by (*rule steps-refl*)
ultimately show *?thesis*
by *blast*
qed

```
lemma stepsClosure-contrasimulation-impl-refl-trans-closure-of-Rel:
  fixes Rel :: ('a × 'a) set
    and Cal :: 'a processCalculus
    and P Q :: 'a
  assumes A1: P ℛ↦<Rel,Cal> Q
      and A2: weak-reduction-contrasimulation Rel Cal
  shows ∃ Q'. Q ⟼Cal∗ Q' ∧ (Q', P) ∈ Rel*
proof −
  have ∀ P'. P ⟼Cal∗ P' ⟶ (∃ Q'. Q ⟼Cal∗ Q' ∧ (Q', P') ∈ Rel*)
    using A1
  proof induct
    case (rel P Q)
    assume (P, Q) ∈ Rel
    with A2 have ∀ P'. P ⟼Cal∗ P' ⟶ (∃ Q'. Q ⟼Cal∗ Q' ∧ (Q', P') ∈ Rel)
      by blast
    thus ∀ P'. P ⟼Cal∗ P' ⟶ (∃ Q'. Q ⟼Cal∗ Q' ∧ (Q', P') ∈ Rel*)
      by blast
  next
    case (steps P P')
    assume A: P ⟼Cal∗ P'
    show ∀ P''. P' ⟼Cal∗ P'' ⟶ (∃ Q'. P ⟼Cal∗ Q' ∧ (Q', P'') ∈ Rel*)
    proof clarify
      fix P''
      assume P' ⟼Cal∗ P''
      with A have P ⟼Cal∗ P''
        by (rule steps-add)
      moreover have (P'', P'') ∈ Rel*
        by simp
      ultimately show ∃ Q'. P ⟼Cal∗ Q' ∧ (Q', P'') ∈ Rel*
        by blast
    qed
  next
    case (trans P Q R)
    assume A1: ∀ P'. P ⟼Cal∗ P' ⟶ (∃ Q'. Q ⟼Cal∗ Q' ∧ (Q', P') ∈ Rel*)
       and A2: ∀ Q'. Q ⟼Cal∗ Q' ⟶ (∃ R'. R ⟼Cal∗ R' ∧ (R', Q') ∈ Rel*)
    show ∀ P'. P ⟼Cal∗ P' ⟶ (∃ R'. R ⟼Cal∗ R' ∧ (R', P') ∈ Rel*)
    proof clarify
      fix P'
      assume P ⟼Cal∗ P'
      with A1 obtain Q' where A3: Q ⟼Cal∗ Q' and A4: (Q', P') ∈ Rel*
        by blast
      from A2 A3 obtain R' where A5: R ⟼Cal∗ R' and A6: (R', Q') ∈ Rel*
        by blast
      from A4 A6 have (R', P') ∈ Rel*
        by simp
      with A5 show ∃ R'. R ⟼Cal∗ R' ∧ (R', P') ∈ Rel*
        by blast
    qed
  qed
  moreover have P ⟼Cal∗ P
    by (rule steps-refl)
  ultimately show ?thesis
    by blast
qed

lemma stepsClosure-contrasimulation-of-inverse-impl-refl-trans-closure-of-Rel:
  fixes Rel :: ('a × 'a) set
    and Cal :: 'a processCalculus
    and P Q :: 'a
  assumes A1: P ℛ↦<Rel⁻¹,Cal> Q
      and A2: weak-reduction-contrasimulation (Rel⁻¹) Cal
```

shows $\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P,\ Q') \in Rel^*$
proof $-$
have $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
using *A1*
proof *induct*
case (*rel P Q*)
assume $(P,\ Q) \in Rel^{-1}$
with *A2* **have** $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (Q',\ P') \in Rel^{-1})$
by *blast*
thus $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
by *blast*
next
case (*steps P P'*)
assume *A*: $P \longmapsto Cal{*}\ P'$
show $\forall\, P''.\ P' \longmapsto Cal{*}\ P'' \longrightarrow (\exists\, Q'.\ P \longmapsto Cal{*}\ Q' \wedge (P'',\ Q') \in Rel^*)$
proof *clarify*
fix P''
assume $P' \longmapsto Cal{*}\ P''$
with *A* **have** $P \longmapsto Cal{*}\ P''$
by (*rule steps-add*)
moreover have $(P'',\ P'') \in Rel^*$
by *simp*
ultimately show $\exists\, Q'.\ P \longmapsto Cal{*}\ Q' \wedge (P'',\ Q') \in Rel^*$
by *blast*
qed
next
case (*trans P Q R*)
assume *A1*: $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, Q'.\ Q \longmapsto Cal{*}\ Q' \wedge (P',\ Q') \in Rel^*)$
and *A2*: $\forall\, Q'.\ Q \longmapsto Cal{*}\ Q' \longrightarrow (\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (Q',\ R') \in Rel^*)$
show $\forall\, P'.\ P \longmapsto Cal{*}\ P' \longrightarrow (\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (P',\ R') \in Rel^*)$
proof *clarify*
fix P'
assume $P \longmapsto Cal{*}\ P'$
with *A1* **obtain** Q' **where** *A3*: $Q \longmapsto Cal{*}\ Q'$ **and** *A4*: $(P',\ Q') \in Rel^*$
by *blast*
from *A3 A2* **obtain** R' **where** *A5*: $R \longmapsto Cal{*}\ R'$ **and** *A6*: $(Q',\ R') \in Rel^*$
by *blast*
from *A4 A6* **have** $(P',\ R') \in Rel^*$
by *simp*
with *A5* **show** $\exists\, R'.\ R \longmapsto Cal{*}\ R' \wedge (P',\ R') \in Rel^*$
by *blast*
qed
qed
moreover have $P \longmapsto Cal{*}\ P$
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
  $STCal\ Source\ Target \equiv$
  $(\!|Reductions = \lambda P\ P'.$
  $(\exists\, SP\ SP'.\ P = SourceTerm\ SP \wedge P' = SourceTerm\ SP' \wedge Reductions\ Source\ SP\ SP') \vee$
  $(\exists\, TP\ TP'.\ P = TargetTerm\ TP \wedge P' = TargetTerm\ TP' \wedge Reductions\ Target\ TP\ TP')|\!)$

definition *STCalWB*
  $:: ('procS, 'barbs)\ calculusWithBarbs \Rightarrow ('procT, 'barbs)\ calculusWithBarbs$
    $\Rightarrow (('procS, 'procT)\ Proc, 'barbs)\ calculusWithBarbs$
  where
  $STCalWB\ Source\ Target \equiv$
  $(\!|Calculus = STCal\ (calculusWithBarbs.Calculus\ Source)\ (calculusWithBarbs.Calculus\ Target),$
  $HasBarb\ \ = \lambda P\ a.\ (\exists\, SP.\ P = SourceTerm\ SP \wedge (calculusWithBarbs.HasBarb\ Source)\ SP\ a) \vee$
  $(\exists\, TP.\ P = TargetTerm\ TP \wedge (calculusWithBarbs.HasBarb\ Target)\ TP\ a)|\!)$

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 $enc :: 'procS \Rightarrow 'procT$ (‹⟦-⟧› [65] 70) **where**
  $[\![S]\!] \equiv Enc\ S$

abbreviation $isSource :: ('procS, 'procT)\ Proc \Rightarrow bool$ (‹- ∈ ProcS› [70] 80) **where**
  $P \in ProcS \equiv (\exists\, S.\ P = SourceTerm\ S)$

abbreviation $isTarget :: ('procS, 'procT)\ Proc \Rightarrow bool$ (‹- ∈ ProcT› [70] 80) **where**
  $P \in ProcT \equiv (\exists\, T.\ P = TargetTerm\ T)$

abbreviation *getSource*
  $:: 'procS \Rightarrow ('procS, 'procT)\ Proc \Rightarrow bool$ (‹- ∈S -› [70, 70] 80)
  where
  $S \in S\ P \equiv (P = SourceTerm\ S)$

abbreviation *getTarget*
  $:: 'procT \Rightarrow ('procS, 'procT)\ Proc \Rightarrow bool$ (‹- ∈T -› [70, 70] 80)
  where
  $T \in T\ P \equiv (P = TargetTerm\ T)$

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

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

lemma *stepST-STCal-step*:
  fixes $P\ P' :: ('procS, 'procT)\ Proc$

```
  shows P ⟼(STCal Source Target) P′ = P ⟼ST P′
    by (simp add: STCal-def)

lemma STStep-step:
  fixes S :: ′procS
    and T :: ′procT
    and P′ :: (′procS, ′procT) Proc
  shows SourceTerm S ⟼ST P′ = (∃ S′. S′ ∈S P′ ∧ S ⟼Source S′)
    and TargetTerm T ⟼ST P′ = (∃ T′. T′ ∈T P′ ∧ T ⟼Target T′)
    by blast+

lemma STCal-step:
  fixes S :: ′procS
    and T :: ′procT
    and P′ :: (′procS, ′procT) Proc
  shows SourceTerm S ⟼(STCal Source Target) P′ = (∃ S′. S′ ∈S P′ ∧ S ⟼Source S′)
    and TargetTerm T ⟼(STCal Source Target) P′ = (∃ T′. T′ ∈T P′ ∧ T ⟼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 ⇒ (′procS, ′procT) Proc ⇒ bool (‹- ⟼ST∗ -› [70, 70] 80)
  where
  P ⟼ST∗ P′ ≡
   (∃ S S′. S ∈S P ∧ S′ ∈S P′ ∧ S ⟼Source∗ S′) ∨ (∃ T T′. T ∈T P ∧ T′ ∈T P′ ∧ T ⟼Target∗ T′)

lemma STSteps-steps:
  fixes S :: ′procS
    and T :: ′procT
    and P′ :: (′procS, ′procT) Proc
  shows SourceTerm S ⟼ST∗ P′ = (∃ S′. S′ ∈S P′ ∧ S ⟼Source∗ S′)
    and TargetTerm T ⟼ST∗ P′ = (∃ T′. T′ ∈T P′ ∧ T ⟼Target∗ T′)
    by blast+

lemma STCal-steps:
  fixes S :: ′procS
    and T :: ′procT
    and P′ :: (′procS, ′procT) Proc
  shows SourceTerm S ⟼(STCal Source Target)∗ P′ = (∃ S′. S′ ∈S P′ ∧ S ⟼Source∗ S′)
    and TargetTerm T ⟼(STCal Source Target)∗ P′ = (∃ T′. T′ ∈T P′ ∧ T ⟼Target∗ T′)
proof auto
  assume SourceTerm S ⟼(STCal Source Target)∗ P′
  from this obtain n where SourceTerm S ⟼(STCal Source Target)^n P′
    by (auto simp add: steps-def)
  thus ∃ S′. S′ ∈S P′ ∧ S ⟼Source∗ S′
  proof (induct n arbitrary: P′)
    case 0
    assume SourceTerm S ⟼(STCal Source Target)^0 P′
    hence S ∈S P′
      by simp
    moreover have S ⟼Source∗ S
      by (rule steps-refl)
    ultimately show ∃ S′. S′ ∈S P′ ∧ S ⟼Source∗ S′
      by blast
  next
    case (Suc n P′′)
    assume SourceTerm S ⟼(STCal Source Target)^(Suc n) P′′
    from this obtain P′ where A1: SourceTerm S ⟼(STCal Source Target)^n P′
                      and A2: P′ ⟼(STCal Source Target) P′′
```

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

assume ⋀*P′*. *TargetTerm T* ⟼(*STCal Source Target*)^{n} *P′* ⟹ ∃ *T′*. *T′* ∈*T P′* ∧ *T* ⟼*Target*∗ *T′*
with *A1* **obtain** *T′* **where** *A3*: *T′* ∈*T P′* **and** *A4*: *T* ⟼*Target*∗ *T′*
by *blast*
from *A2 A3* **obtain** *T″* **where** *A5*: *T″* ∈*T P″* **and** *A6*: *T′* ⟼*Target T″*
using *STCal-step*(*2*)[**where** *T*=*T′* **and** *P′*=*P″*]
by *blast*
from *A4 A6* **have** *T* ⟼*Target*∗ *T″*
using *step-to-steps*[**where** *Cal*=*Target* **and** *P*=*T′* **and** *P′*=*T″*]
by (*simp add*: *steps-add*[**where** *Cal*=*Target* **and** *P*=*T* **and** *Q*=*T′* **and** *R*=*T″*])
with *A5* **show** ∃ *T″*. *T″* ∈*T P″* ∧ *T* ⟼*Target*∗ *T″*
by *blast*
qed
next
fix *T′*
assume *T* ⟼*Target*∗ *T′*
from *this* **obtain** *n* **where** *T* ⟼*Target*^{n} *T′*
by (*auto simp add*: *steps-def*)
thus *TargetTerm T* ⟼(*STCal Source Target*)∗ (*TargetTerm T′*)
proof (*induct n arbitrary*: *T′*)
case *0*
assume *T* ⟼*Target*^{0} *T′*
hence *T* = *T′*
by *auto*
thus *TargetTerm T* ⟼(*STCal Source Target*)∗ (*TargetTerm T′*)
by (*simp add*: *steps-refl*)
next
case (*Suc n T″*)
assume *T* ⟼*Target*$^{Suc\ n}$ *T″*
from *this* **obtain** *T′* **where** *B1*: *T* ⟼*Target*^{n} *T′* **and** *B2*: *T′* ⟼*Target T″*
by *auto*
assume ⋀*T′*. *T* ⟼*Target*^{n} *T′* ⟹ *TargetTerm T* ⟼(*STCal Source Target*)∗ (*TargetTerm T′*)
with *B1* **have** *TargetTerm T* ⟼(*STCal Source Target*)∗ (*TargetTerm T′*)
by *blast*
moreover from *B2* **have** *TargetTerm T′* ⟼(*STCal Source Target*)∗ (*TargetTerm T″*)
using *step-to-steps*[**where** *Cal*=*STCal Source Target* **and** *P*=*TargetTerm T′*]
by (*simp add*: *STCal-def*)
ultimately show *TargetTerm T* ⟼(*STCal Source Target*)∗ (*TargetTerm T″*)
by (*rule steps-add*)
qed
qed

lemma *stepsST-STCal-steps*:
fixes *P P′* :: (*′procS*, *′procT*) *Proc*
shows *P* ⟼(*STCal Source Target*)∗ *P′* = *P* ⟼*ST*∗ *P′*
proof (*cases P*)
case (*SourceTerm SP*)
assume *SP* ∈*S P*
thus *P* ⟼(*STCal Source Target*)∗ *P′* = *P* ⟼*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* ∈*T P*
thus *P* ⟼(*STCal Source Target*)∗ *P′* = *P* ⟼*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 $P \longmapsto ST* \ P$
by (*cases P*, *simp-all add*: *steps-refl*)

lemma *stepsST-add*:
fixes $P\ Q\ R :: ('procS, 'procT)\ Proc$
assumes *A1*: $P \longmapsto ST* \ Q$
and *A2*: $Q \longmapsto ST* \ R$
shows $P \longmapsto ST* \ R$
proof −
from *A1* **have** $P \longmapsto (STCal\ Source\ Target)* \ Q$
by (*simp add*: *stepsST-STCal-steps*)
moreover from *A2* **have** $Q \longmapsto (STCal\ Source\ Target)* \ R$
by (*simp add*: *stepsST-STCal-steps*)
ultimately have $P \longmapsto (STCal\ Source\ Target)* \ R$
by (*rule steps-add*)
thus $P \longmapsto ST* \ R$
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*
$:: ('procS, 'procT)\ Proc \Rightarrow bool$ (‹- ⟼STω› [*70*] *80*)
where
$P \longmapsto ST\omega \equiv (\exists S.\ S \in S\ P \wedge S \longmapsto (Source)\omega) \vee (\exists T.\ T \in T\ P \wedge T \longmapsto (Target)\omega)$

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

lemma *divergentST-STCal-divergent*:
fixes $P :: ('procS, 'procT)\ Proc$
shows $P \longmapsto (STCal\ Source\ Target)\omega = P \longmapsto ST\omega$
proof (*cases P*)
case (*SourceTerm SP*)
assume $SP \in S\ P$
thus $P \longmapsto (STCal\ Source\ Target)\omega = P \longmapsto ST\omega$
using *STCal-divergent*(*1*)
by *simp*
next
case (*TargetTerm TP*)
assume $TP \in T\ P$
thus $P \longmapsto (STCal\ Source\ Target)\omega = P \longmapsto ST\omega$
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**
$enc\text{-}preserves\text{-}pred\ Pred \equiv \forall S.\ Pred\ (SourceTerm\ S) \longrightarrow Pred\ (TargetTerm\ ([\![S]\!]))$

abbreviation *enc-preserves-binary-pred*
$:: (('procS, 'procT)\ Proc \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$
where
$enc\text{-}preserves\text{-}binary\text{-}pred\ Pred \equiv \forall S\ x.\ Pred\ (SourceTerm\ S)\ x \longrightarrow Pred\ (TargetTerm\ ([\![S]\!]))\ x$

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

abbreviation *enc-reflects-pred* :: $(('procS, 'procT)\ Proc \Rightarrow bool) \Rightarrow bool$ **where**
enc-reflects-pred Pred $\equiv \forall S.\ Pred\ (TargetTerm\ ([\![S]\!])) \longrightarrow Pred\ (SourceTerm\ S)$

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

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

abbreviation *enc-respects-pred* :: $(('procS, 'procT)\ Proc \Rightarrow bool) \Rightarrow bool$ **where**
enc-respects-pred Pred $\equiv$ *enc-preserves-pred Pred* $\wedge$ *enc-reflects-pred Pred*

abbreviation *enc-respects-binary-pred*
:: $(('procS, 'procT)\ Proc \Rightarrow 'b \Rightarrow bool) \Rightarrow bool$
where
enc-respects-binary-pred Pred $\equiv$
enc-preserves-binary-pred Pred $\wedge$ *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* $\Rightarrow$ *'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*
:: $('procS, 'procT)\ Proc \Rightarrow 'barbs \Rightarrow bool$ (‹-↓.-› [70, 70] 80)
where
$P{\downarrow}.a \equiv (\exists S.\ S \in S\ P \wedge S{\downarrow}\text{<}SWB\text{>}a) \vee (\exists T.\ T \in T\ P \wedge T{\downarrow}\text{<}TWB\text{>}a)$

lemma *STCalWB-hasBarbST*:
fixes *P* :: *('procS, 'procT) Proc*
and *a* :: *'barbs*
shows $P{\downarrow}\text{<}STCalWB\ SWB\ TWB\text{>}a = P{\downarrow}.a$
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*: $(P, Q) \in Rel$
and *barb*: $P{\downarrow}.a$
shows $Q{\downarrow}.a$

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

lemma reflection-of-barbs-in-barbed-encoding:
 fixes Rel :: (('procS, 'procT) Proc × ('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) ∈ Rel
    and barb:        Q↓.a
 shows P↓.a
    using reflection rel barb
  by (simp add: STCalWB-def)

lemma respection-of-barbs-in-barbed-encoding:
 fixes Rel :: (('procS, 'procT) Proc × ('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) ∈ Rel
 shows P↓.a = Q↓.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 ⇒ 'barbs ⇒ bool (‹-⇓.-› [70, 70] 80)
 where
 P⇓.a ≡ (∃ S. S ∈S P ∧ S⇓<SWB>a) ∨ (∃ T. T ∈T P ∧ T⇓<TWB>a)

lemma STCalWB-reachesBarbST:
 fixes P :: ('procS, 'procT) Proc
  and a :: 'barbs
 shows P⇓<STCalWB SWB TWB>a = P⇓.a
proof −
 have ∀ S. SourceTerm S⇓<STCalWB SWB TWB>a = SourceTerm S⇓.a
    using STCal-steps(1)
  by (auto simp add: STCalWB-def calS calT)
 moreover have ∀ T. TargetTerm T⇓<STCalWB SWB TWB>a = TargetTerm T⇓.a
    using STCal-steps(2)
  by (auto simp add: STCalWB-def calS calT)
 ultimately show P⇓<STCalWB SWB TWB>a = P⇓.a
  by (cases P, simp+)
qed

lemma weak-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-weakly-preserves-barbs Rel (STCalWB SWB TWB)
    and rel:          (P, Q) ∈ Rel
    and barb:          P⇓.a
 shows Q⇓.a
proof −
 from barb have P⇓<STCalWB SWB TWB>a
  by (simp add: STCalWB-reachesBarbST)
 with preservation rel have Q⇓<STCalWB SWB TWB>a
```

```
    by blast
  thus Q⇓.a
    by (simp add: STCalWB-reachesBarbST)
qed

lemma weak-reflection-of-barbs-in-barbed-encoding:
  fixes Rel :: (('procS, 'procT) Proc × ('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) ∈ Rel
      and barb:        Q⇓.a
  shows P⇓.a
proof −
  from barb have Q⇓<STCalWB SWB TWB>a
    by (simp add: STCalWB-reachesBarbST)
  with reflection rel have P⇓<STCalWB SWB TWB>a
    by blast
  thus P⇓.a
    by (simp add: STCalWB-reachesBarbST)
qed

lemma weak-respection-of-barbs-in-barbed-encoding:
  fixes Rel :: (('procS, 'procT) Proc × ('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) ∈ Rel
  shows P⇓.a = Q⇓.a
proof (rule iffI)
  assume P⇓.a
  with respection rel show Q⇓.a
      using weak-preservation-of-barbs-in-barbed-encoding[where Rel=Rel]
    by blast
next
  assume Q⇓.a
  with respection rel show P⇓.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, enc S) for all source terms S.

```
inductive-set (in encoding) indRelR
    :: ((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
  where
  encR: (SourceTerm S, TargetTerm (⟦S⟧)) ∈ 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: $(SourceTerm\ S,\ TargetTerm\ ([\![S]\!])) \in indRelRPO$ |
source: $(SourceTerm\ S,\ SourceTerm\ S) \in indRelRPO$ |
target: $(TargetTerm\ T,\ TargetTerm\ T) \in indRelRPO$ |
trans: $[\![(P,\ Q) \in indRelRPO;\ (Q,\ R) \in indRelRPO]\!] \Longrightarrow (P,\ R) \in indRelRPO$

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* ($[\![S]\!]$)) $\in$ *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* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ *PredA Rel*
and $\forall (P, Q) \in$ *indRelR*. *PredB* (P, Q)
$\Longrightarrow \exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ ($\forall (P, Q) \in$ *Rel*. *PredB* (P, Q))
proof –
have *A*: $\forall$ *S*. *SourceTerm S* $\mathcal{R}[\![\cdot]\!]R$ *TargetTerm* ($[\![S]\!]$)
by (*simp add*: *indRelR.encR*)
thus *PredA indRelR* $\Longrightarrow \exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *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* ($[\![S]\!]$)) $\in$ *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*)

```
      ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel)
proof (rule iffI)
  have ∀ S. SourceTerm S ≲⟦·⟧R TargetTerm (⟦S⟧)
    by (simp add: indRelRPO.encR)
  moreover have preorder indRelRPO
      using indRelRPO-is-preorder
    by blast
  moreover assume ∀ (P, Q) ∈ indRelRPO. Pred (P, Q)
  ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                    ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
        ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel
  from this obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
                        and A2: ∀ (P, Q) ∈ Rel. Pred (P, Q) and A3: preorder Rel
    by blast
  show ∀ (P, Q) ∈ indRelRPO. Pred (P, Q)
  proof clarify
    fix P Q
    assume P ≲⟦·⟧R Q
    hence (P, Q) ∈ Rel
    proof induct
      case (encR S)
      from A1 show (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
        by simp
    next
      case (source S)
      from A3 show (SourceTerm S, SourceTerm S) ∈ Rel
          unfolding preorder-on-def refl-on-def
        by simp
    next
      case (target T)
      from A3 show (TargetTerm T, TargetTerm T) ∈ Rel
          unfolding preorder-on-def refl-on-def
        by simp
    next
      case (trans P Q R)
      assume (P, Q) ∈ Rel and (Q, R) ∈ Rel
      with A3 show (P, R) ∈ Rel
          unfolding preorder-on-def trans-def
        by blast
    qed
    with A2 show 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 :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
  assumes encCond: ∀ S. Pred (SourceTerm S, TargetTerm (⟦S⟧))
  shows ∀ (P, Q) ∈ indRelR. Pred (P, Q)
    by (auto simp add: encCond indRelR.simps)

lemma (in encoding) indRelR-satisfies-pred-impl-enc-satisfies-pred:
  fixes Pred :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) ⇒ bool
  assumes relCond: ∀ (P, Q) ∈ indRelR. Pred (P, Q)
  shows ∀ S. Pred (SourceTerm S, TargetTerm (⟦S⟧))
```

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-preserves-pred Pred = rel-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-preserves-binary-pred Pred = rel-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-preserves-pred Pred = rel-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-reflects-pred Pred = rel-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-reflects-binary-pred Pred = rel-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-reflects-pred Pred = rel-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-respects-pred Pred = rel-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*=$\lambda(P, Q)$. *Pred* Q = *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 ($\forall S$. *Pred* (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)))
= ($\exists$ *Rel*. ($\forall S$. (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ ($\forall (P, Q) \in$ *Rel*. *Pred* (P, Q)))
and $[\![\forall P\ Q\ R.$ *Pred* $(P, Q) \wedge$ *Pred* $(Q, R) \longrightarrow$ *Pred* (P, R); $\forall P$. *Pred* $(P, P)]\!] \Longrightarrow$
($\forall S$. *Pred* (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$))) = ($\exists$ *Rel*. ($\forall S$.
(*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ ($\forall (P, Q) \in$ *Rel*. *Pred* (P, Q)) $\wedge$ *preorder Rel*)
proof −
show ($\forall S$. *Pred* (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)))
= ($\exists$ *Rel*. ($\forall S$. (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ ($\forall (P, Q) \in$ *Rel*. *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 ($\forall S$. *Pred* (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$))) = ($\forall (P, Q) \in$ *indRelR*. *Pred* (P, Q))
using *enc-satisfies-pred-iff-indRelR-satisfies-pred*[**where** *Pred*=*Pred*]
by *simp*
moreover assume $\forall P\ Q\ R$. *Pred* $(P, Q) \wedge$ *Pred* $(Q, R) \longrightarrow$ *Pred* (P, R) **and** $\forall P$. *Pred* (P, P)
hence ($\forall (P, Q) \in$ *indRelR*. *Pred* (P, Q)) = ($\forall (P, Q) \in$ *indRelRPO*. *Pred* (P, Q))
using *indRelR-modulo-pred-impl-indRelRPO-modulo-pred*[**where** *Pred*=*Pred*]
by *blast*
ultimately show ($\forall S$. *Pred* (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$))) = ($\exists$ *Rel*.
($\forall S$. (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ ($\forall (P, Q) \in$ *Rel*. *Pred* (P, Q)) $\wedge$ *preorder 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*
= ($\exists$ *Rel*. ($\forall S$. (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*) $\wedge$ *rel-preserves-pred Rel Pred*)
and *enc-preserves-pred Pred* = ($\exists$ *Rel*. ($\forall S$. (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ *rel-preserves-pred Rel Pred* $\wedge$ *preorder Rel*)
proof −
have *A1*: *enc-preserves-pred Pred*
= ($\forall S$. ($\lambda(P, Q)$. *Pred* $P \longrightarrow$ *Pred* Q) (*SourceTerm* S, *TargetTerm* ($[\![S]\!]$)))
by *blast*
moreover have *A2*: $\bigwedge$*Rel*. *rel-preserves-pred Rel Pred*
= ($\forall (P, Q) \in$ *Rel*. ($\lambda(P, Q)$. *Pred* $P \longrightarrow$ *Pred* Q) (P, Q))
by *blast*

ultimately show *enc-preserves-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-preserves-pred Rel Pred*)
using *enc-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-preserves-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-preserves-pred Rel Pred* $\wedge$ *preorder Rel*)
using *enc-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-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow\ 'b \Rightarrow bool$
shows *enc-preserves-binary-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-preserves-binary-pred Rel Pred*)
proof –
have *enc-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-preserves-binary-pred Rel 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-preserves-binary-pred Pred* $= (\exists\, Rel.\ (\forall\, S.$
$(SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge$ *rel-preserves-binary-pred Rel Pred*)
using *enc-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-reflects-pred-iff-source-target-rel-reflects-pred*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow bool$
shows *enc-reflects-pred Pred*
$= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge$ *rel-reflects-pred Rel Pred*)
and *enc-reflects-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-reflects-pred Rel Pred* $\wedge$ *preorder Rel*)
proof –
have *A1*: *enc-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-reflects-pred Rel Pred*
$= (\forall\, (P,\ Q) \in Rel.\ (\lambda(P,\ Q).\ Pred\ Q \longrightarrow Pred\ P)\ (P,\ Q))$
by *blast*
ultimately show *enc-reflects-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-reflects-pred Rel Pred*)
using *enc-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-reflects-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-reflects-pred Rel Pred* $\wedge$ *preorder Rel*)
using *enc-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-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow\ 'b \Rightarrow bool$
shows *enc-reflects-binary-pred Pred* $= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge$ *rel-reflects-binary-pred Rel Pred*)
proof –
have *enc-reflects-binary-pred Pred*

$= (\forall S.\ (\lambda(P,\ Q).\ \forall a.\ Pred\ Q\ a \longrightarrow Pred\ P\ a)\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)))$
by *blast*
moreover have $\bigwedge Rel.\ rel\text{-}reflects\text{-}binary\text{-}pred\ Rel\ Pred$
$= (\forall (P,\ Q) \in Rel.\ (\lambda(P,\ Q).\ \forall a.\ Pred\ Q\ a \longrightarrow Pred\ P\ a)\ (P,\ Q))$
by *blast*
ultimately show $enc\text{-}reflects\text{-}binary\text{-}pred\ Pred = (\exists Rel.\ (\forall S.$
$(SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \land rel\text{-}reflects\text{-}binary\text{-}pred\ Rel\ Pred)$
using $enc\text{-}satisfies\text{-}pred\text{-}iff\text{-}source\text{-}target\text{-}satisfies\text{-}pred(1)$[**where**
$Pred{=}\lambda(P,\ Q).\ \forall a.\ Pred\ Q\ a \longrightarrow Pred\ P\ a$]
by *simp*
qed

lemma (**in** *encoding*) *enc-respects-pred-iff-source-target-rel-respects-pred-encR*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow bool$
shows $enc\text{-}respects\text{-}pred\ Pred$
$= (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \land rel\text{-}respects\text{-}pred\ Rel\ Pred)$
and $enc\text{-}respects\text{-}pred\ Pred = (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\land\ rel\text{-}respects\text{-}pred\ Rel\ Pred \land preorder\ Rel)$
proof −
have $A1$: $enc\text{-}respects\text{-}pred\ Pred$
$= (\forall S.\ (\lambda(P,\ Q).\ Pred\ P = Pred\ Q)\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)))$
by *blast*
moreover
have $A2$: $\bigwedge Rel.\ rel\text{-}respects\text{-}pred\ Rel\ Pred = (\forall (P,\ Q) \in Rel.\ (\lambda(P,\ Q).\ Pred\ P = Pred\ Q)\ (P,\ Q))$
by *blast*
ultimately show $enc\text{-}respects\text{-}pred\ Pred = (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\land\ rel\text{-}respects\text{-}pred\ Rel\ Pred)$
using $enc\text{-}satisfies\text{-}pred\text{-}iff\text{-}source\text{-}target\text{-}satisfies\text{-}pred(1)$[**where**
$Pred{=}\lambda(P,\ Q).\ Pred\ P = Pred\ Q$]
by *simp*
from $A1\ A2$ **show** $enc\text{-}respects\text{-}pred\ Pred = (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\land\ rel\text{-}respects\text{-}pred\ Rel\ Pred \land preorder\ Rel)$
using $enc\text{-}satisfies\text{-}pred\text{-}iff\text{-}source\text{-}target\text{-}satisfies\text{-}pred(2)$[**where**
$Pred{=}\lambda(P,\ Q).\ Pred\ P = Pred\ Q$]
by *simp*
qed

lemma (**in** *encoding*) *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow 'b \Rightarrow bool$
shows $enc\text{-}respects\text{-}binary\text{-}pred\ Pred = (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\land\ rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred)$
proof −
have $enc\text{-}respects\text{-}binary\text{-}pred\ Pred$
$= (\forall S.\ (\lambda(P,\ Q).\ \forall a.\ Pred\ P\ a = Pred\ Q\ a)\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)))$
by *blast*
moreover have $\bigwedge Rel.\ rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred$
$= (\forall (P,\ Q) \in Rel.\ (\lambda(P,\ Q).\ \forall a.\ Pred\ P\ a = Pred\ Q\ a)\ (P,\ Q))$
by *blast*
ultimately show $enc\text{-}respects\text{-}binary\text{-}pred\ Pred = (\exists Rel.\ (\forall S.$
$(SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \land rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred)$
using $enc\text{-}satisfies\text{-}pred\text{-}iff\text{-}source\text{-}target\text{-}satisfies\text{-}pred(1)$[**where**
$Pred{=}\lambda(P,\ Q).\ \forall a.\ Pred\ P\ a = Pred\ Q\ a$]
by *simp*
qed

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

inductive-set (**in** *encoding*) *indRelL*
$:: ((('procS,\ 'procT)\ Proc) \times (('procS,\ 'procT)\ Proc))\ set$
where

encL: (*TargetTerm* ($[\![S]\!]$), *SourceTerm* S) $\in$ *indRelL*

abbreviation (**in** *encoding*) *indRelLinfix* ::
('*procS*, '*procT*) *Proc* $\Rightarrow$ ('*procS*, '*procT*) *Proc* $\Rightarrow$ *bool* (‹- $\mathcal{R}[\![\cdot]\!]L$ -› [*75*, *75*] *80*)
where
$P\ \mathcal{R}[\![\cdot]\!]L\ Q \equiv (P,\ Q) \in$ *indRelL*

inductive-set (**in** *encoding*) *indRelLPO*
:: ((('*procS*, '*procT*) *Proc*) $\times$ (('*procS*, '*procT*) *Proc*)) *set*
where
encL: (*TargetTerm* ($[\![S]\!]$), *SourceTerm* S) $\in$ *indRelLPO* |
source: (*SourceTerm* S, *SourceTerm* S) $\in$ *indRelLPO* |
target: (*TargetTerm* T, *TargetTerm* T) $\in$ *indRelLPO* |
trans: $[\![(P,\ Q) \in$ *indRelLPO*; $(Q,\ R) \in$ *indRelLPO*$]\!] \Longrightarrow (P,\ R) \in$ *indRelLPO*

abbreviation (**in** *encoding*) *indRelLPOinfix* ::
('*procS*, '*procT*) *Proc* $\Rightarrow$ ('*procS*, '*procT*) *Proc* $\Rightarrow$ *bool* (‹- $\lesssim[\![\cdot]\!]L$ -› [*75*, *75*] *80*)
where
$P \lesssim[\![\cdot]\!]L\ Q \equiv (P,\ Q) \in$ *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 $(P,\ Q) \in$ *indRelL**
proof *induct*
case (*encL S*)
show (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) $\in$ *indRelL**
using *indRelL.encL*[*of S*]
by *simp*
next
case (*source S*)
show (*SourceTerm S*, *SourceTerm S*) $\in$ *indRelL**
by *simp*
next
case (*target T*)
show (*TargetTerm T*, *TargetTerm T*) $\in$ *indRelL**
by *simp*
next
case (*trans P Q R*)
assume $(P,\ Q) \in$ *indRelL** **and** $(Q,\ R) \in$ *indRelL**
thus $(P,\ R) \in$ *indRelL**
by *simp*
qed
next
fix *P Q*
assume $(P,\ Q) \in$ *indRelL**
thus $P \lesssim[\![\cdot]\!]L\ Q$
proof *induct*
show $P \lesssim[\![\cdot]\!]L\ P$
using *indRelLPO-refl*
unfolding *refl-on-def*
by *simp*
next
case (*step Q R*)
assume $P \lesssim[\![\cdot]\!]L\ Q$
moreover assume $Q\ \mathcal{R}[\![\cdot]\!]L\ R$
hence $Q \lesssim[\![\cdot]\!]L\ R$
by (*induct*, *simp add*: *indRelLPO.encL*)
ultimately show $P \lesssim[\![\cdot]\!]L\ R$
by (*simp add*: *indRelLPO.trans*[*of P Q R*])
qed
qed

The relations indRelR and indRelL are dual. indRelR preserves some predicate iff indRelL reflects it. indRelR reflects some predicate iff indRelL reflects it. indRelR respects some predicate iff indRelL does.

lemma (**in** *encoding*) *indRelR-preserves-pred-iff-indRelL-reflects-pred*:
fixes *Pred* :: (*'procS*, *'procT*) *Proc* $\Rightarrow$ *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 $P\ \mathcal{R}[\![\cdot]\!]L\ Q$
from *this* **obtain** *S* **where** $S \in S\ Q$ **and** $[\![S]\!] \in T\ P$
by (*induct*, *blast*)
hence $Q\ \mathcal{R}[\![\cdot]\!]R\ P$
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** $[\![S]\!] \in T\ Q$
      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** $[\![S]\!] \in T\ P$
      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** $[\![S]\!] \in T\ Q$
      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** $[\![S]\!] \in T\ P$
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** $[\![S]\!] \in T\ Q$
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** $[\![S]\!] \in T\ P$
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** $[\![S]\!] \in T\ Q$
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* ⇒ *bool*
  shows *rel-respects-pred indRelR Pred* = *rel-respects-pred indRelL Pred*
    using *indRelR-preserves-pred-iff-indRelL-reflects-pred*[**where** *Pred*=*Pred*]
      indRelR-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* ⇒'*b* ⇒ *bool*
  shows *rel-respects-binary-pred indRelR Pred* = *rel-respects-binary-pred indRelL Pred*
    using *indRelR-preserves-binary-pred-iff-indRelL-reflects-binary-pred*[**where** *Pred*=*Pred*]
      indRelR-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* ⇒ *bool*
  shows (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*)
    = (∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*)
proof
  assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*
  then obtain *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
      and *A2*: *rel-preserves-pred Rel Pred*
    by *blast*
  from *A1* **have** ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ Rel^{-1}
    by *simp*
  moreover from *A2* **have** *rel-reflects-pred* (Rel^{-1}) *Pred*
    by *simp*
  ultimately show ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*
    by *blast*
next
  assume ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*
  then obtain *Rel* **where** *B1*: ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
      and *B2*: *rel-reflects-pred Rel Pred*
    by *blast*
  from *B1* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ Rel^{-1}
    by *simp*
  moreover from *B2* **have** *rel-preserves-pred* (Rel^{-1}) *Pred*
    by *blast*
  ultimately
  show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*
    by *blast*
qed

lemma (**in** *encoding*) *indRelR-cond-binary-preservation-iff-indRelL-cond-binary-reflection*:
  fixes *Pred* :: ('*procS*, '*procT*) *Proc* ⇒ '*b* ⇒ *bool*
  shows (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-binary-pred Rel Pred*)
    = (∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
      ∧ *rel-reflects-binary-pred Rel Pred*)
proof
  assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-binary-pred Rel Pred*
  then obtain *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
      and *A2*: *rel-preserves-binary-pred Rel Pred*
    by *blast*
  from *A1* **have** ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ Rel^{-1}
    by *simp*
  moreover from *A2* **have** *rel-reflects-binary-pred* (Rel^{-1}) *Pred*
    by *simp*
  ultimately
  show ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-reflects-binary-pred Rel Pred*
    by *blast*

next
assume ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-reflects-binary-pred Rel Pred*
then obtain *Rel* **where** *B1*: ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
and *B2*: *rel-reflects-binary-pred Rel Pred*
by *blast*
from *B1* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ Rel^{-1}
by *simp*
moreover from *B2* **have** *rel-preserves-binary-pred* (Rel^{-1}) *Pred*
by *simp*
ultimately
show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-preserves-binary-pred Rel Pred*
by *blast*
qed

lemma (**in** *encoding*) *indRelR-cond-reflection-iff-indRelL-cond-preservation*:
fixes *Pred* :: ('*procS*, '*procT*) *Proc* ⇒ *bool*
shows (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*)
= (∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*)
proof
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*
then obtain *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: *rel-reflects-pred Rel Pred*
by *blast*
from *A1* **have** ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ Rel^{-1}
by *simp*
moreover from *A2* **have** *rel-preserves-pred* (Rel^{-1}) *Pred*
by *blast*
ultimately
show ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-preserves-pred Rel Pred*
then obtain *Rel* **where** *B1*: ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
and *B2*: *rel-preserves-pred Rel Pred*
by *blast*
from *B1* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ Rel^{-1}
by *simp*
moreover from *B2* **have** *rel-reflects-pred* (Rel^{-1}) *Pred*
by *simp*
ultimately
show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-reflects-pred Rel Pred*
by *blast*
qed

lemma (**in** *encoding*) *indRelR-cond-binary-reflection-iff-indRelL-cond-binary-preservation*:
fixes *Pred* :: ('*procS*, '*procT*) *Proc* ⇒ '*b* ⇒ *bool*
shows (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-reflects-binary-pred Rel Pred*)
= (∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *rel-preserves-binary-pred Rel Pred*)
proof
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-reflects-binary-pred Rel Pred*
then obtain *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: *rel-reflects-binary-pred Rel Pred*
by *blast*
from *A1* **have** ∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ Rel^{-1}
by *simp*
moreover from *A2* **have** *rel-preserves-binary-pred* (Rel^{-1}) *Pred*
by *blast*
ultimately
show ∃ *Rel*. (∀ *S*. (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-preserves-binary-pred Rel Pred*
by *blast*

next
assume $\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}preserves\text{-}binary\text{-}pred\ Rel\ Pred$
then obtain *Rel* **where** $B1$: $\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel$
and $B2$: $rel\text{-}preserves\text{-}binary\text{-}pred\ Rel\ Pred$
by *blast*
from *B1* **have** $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel^{-1}$
by *simp*
moreover from *B2* **have** $rel\text{-}reflects\text{-}binary\text{-}pred\ (Rel^{-1})\ Pred$
by *simp*
ultimately
show $\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}reflects\text{-}binary\text{-}pred\ Rel\ Pred$
by *blast*
qed

lemma (**in** *encoding*) *indRelR-cond-respection-iff-indRelL-cond-respection*:
fixes $Pred :: ('procS,\ 'procT)\ Proc \Rightarrow bool$
shows $(\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred)$
$= (\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred)$
proof
assume $\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred$
from *this* **obtain** *Rel* **where** $A1$: $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel$
and $A2$: $rel\text{-}respects\text{-}pred\ Rel\ Pred$
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** $rel\text{-}respects\text{-}pred\ \{(a,\ b).\ (b,\ a) \in Rel\}\ Pred$
by *blast*
ultimately show $\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred$
by *blast*
next
assume $\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred$
from *this* **obtain** *Rel* **where** $A1$: $\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel$
and $A2$: $rel\text{-}respects\text{-}pred\ Rel\ Pred$
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** $rel\text{-}respects\text{-}pred\ \{(a,\ b).\ (b,\ a) \in Rel\}\ Pred$
by *blast*
ultimately show $\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}respects\text{-}pred\ Rel\ Pred$
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 $(\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred)$
$= (\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel)$
$\wedge\ rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred)$
proof
assume $\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel) \wedge rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred$
from *this* **obtain** *Rel* **where** $A1$: $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel$
and $A2$: $rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred$
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** $rel\text{-}respects\text{-}binary\text{-}pred\ \{(a,\ b).\ (b,\ a) \in Rel\}\ Pred$
by *blast*
ultimately
show $\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred$
by *blast*
next
assume $\exists Rel.\ (\forall S.\ (TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel) \wedge rel\text{-}respects\text{-}binary\text{-}pred\ Rel\ Pred$

from *this* **obtain** *Rel* **where** *A1*: $\forall S.$ (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) $\in$ *Rel*
and *A2*: *rel-respects-binary-pred Rel Pred*
by *blast*
from *A1* **have** $\forall S.$ (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ {(*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* ($[\![S]\!]$)) $\in$ *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* ($[\![S]\!]$), *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* ($[\![S]\!]$), *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* ($[\![S]\!]$), *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 respection of source term behaviours we use relations that contain both kind of pairs: (S, enc S) as well as (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 (⟦S⟧)) ∈ indRel |
  encL: (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRel

abbreviation (in encoding) indRelInfix ::
    ('procS, 'procT) Proc ⇒ ('procS, 'procT) Proc ⇒ bool (‹- ℛ⟦·⟧ -› [75, 75] 80)
  where
  P ℛ⟦·⟧ 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 (⟦S⟧)) ∈ indRelEQ |
  encL:   (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRelEQ |
  target: (TargetTerm T, TargetTerm T) ∈ indRelEQ |
  trans:  ⟦(P, Q) ∈ indRelEQ; (Q, R) ∈ indRelEQ⟧ ⟹ (P, R) ∈ indRelEQ

abbreviation (in encoding) indRelEQinfix ::
    ('procS, 'procT) Proc ⇒ ('procS, 'procT) Proc ⇒ bool (‹- ∼⟦·⟧ -› [75, 75] 80)
  where
  P ∼⟦·⟧ Q ≡ (P, Q) ∈ indRelEQ

lemma (in encoding) indRelEQ-refl:
  shows refl indRelEQ
    unfolding refl-on-def
proof auto
  fix P
  show P ∼⟦·⟧ P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈S P
    moreover have SourceTerm SP ∼⟦·⟧ TargetTerm (⟦SP⟧)
      by (rule indRelEQ.encR)
    moreover have TargetTerm (⟦SP⟧) ∼⟦·⟧ SourceTerm SP
      by (rule indRelEQ.encL)
    ultimately show P ∼⟦·⟧ P
      by (simp add: indRelEQ.trans[where P=SourceTerm SP and Q=TargetTerm (⟦SP⟧)])
  next
    case (TargetTerm TP)
    assume TP ∈T P
    thus P ∼⟦·⟧ 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)
```

next
  show *trans indRelEQ*
    unfolding *trans-def*
  proof *clarify*
    fix P Q R
    assume $P \sim[\![\cdot]\!]\ Q$ **and** $Q \sim[\![\cdot]\!]\ R$
    thus $P \sim[\![\cdot]\!]\ R$
      by (*rule indRelEQ.trans*)
  qed
qed

lemma (**in** *encoding*) *indRelEQ-symm*:
  shows *sym indRelEQ*
    unfolding *sym-def*
proof *clarify*
  fix P Q
  assume $P \sim[\![\cdot]\!]\ Q$
  thus $Q \sim[\![\cdot]\!]\ P$
  proof *induct*
    case (*encR S*)
    show *TargetTerm* ($[\![S]\!]$) $\sim[\![\cdot]\!]$ *SourceTerm S*
      by (*rule indRelEQ.encL*)
  next
    case (*encL S*)
    show *SourceTerm S* $\sim[\![\cdot]\!]$ *TargetTerm* ($[\![S]\!]$)
      by (*rule indRelEQ.encR*)
  next
    case (*target T*)
    show *TargetTerm T* $\sim[\![\cdot]\!]$ *TargetTerm T*
      by (*rule indRelEQ.target*)
  next
    case (*trans P Q R*)
    assume $R \sim[\![\cdot]\!]\ Q$ **and** $Q \sim[\![\cdot]\!]\ P$
    thus $R \sim[\![\cdot]\!]\ P$
      by (*rule indRelEQ.trans*)
  qed
qed

lemma (**in** *encoding*) *indRelEQ-is-equivalence*:
  shows *equivalence indRelEQ*
      using *indRelEQ-is-preorder indRelEQ-symm*
      unfolding *equiv-def preorder-on-def*
    by *blast*

lemma (**in** *encoding*) *refl-trans-closure-of-indRel*:
  shows *indRelEQ* = *indRel**
proof *auto*
  fix P Q
  assume $P \sim[\![\cdot]\!]\ Q$
  thus $(P, Q) \in$ *indRel**
  proof *induct*
    case (*encR S*)
    show (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *indRel**
        using *indRel.encR*[*of S*]
      by *simp*
  next
    case (*encL S*)
    show (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) $\in$ *indRel**
        using *indRel.encL*[*of S*]
      by *simp*
  next

```
    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 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 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 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*) ∈ *symcl indRelL*
     by (*induct*, *simp-all add*: *symcl-def indRelL.encL*)
  next
    fix *P Q*
    assume (*P*, *Q*) ∈ *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* ⇒ *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* ∈*S* *P* **and** $[\![S]\!]$ ∈*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* ∈*S* *P* **and** $[\![S]\!]$ ∈*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* ⇒ '*b* ⇒ *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** $[\![S]\!] \in T\ Q$
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** $[\![S]\!] \in T\ Q$
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* ⇒ *bool*
shows (∃ *Rel*.
(∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel* ∧ (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) ∈ *Rel*)
∧ *rel-respects-pred Rel Pred*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*) ∧ *rel-respects-pred Rel Pred*)
proof
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*
∧ (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) ∈ *Rel*) ∧ *rel-respects-pred Rel Pred*
from *this* **obtain** *Rel*
where ∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel* ∧ (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) ∈ *Rel*
and *rel-respects-pred Rel Pred*
by *blast*
thus ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*) ∧ *rel-respects-pred Rel Pred*
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*) ∧ *rel-respects-pred Rel Pred*
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*
and *A2*: *rel-respects-pred Rel Pred*
by *blast*
from *A1* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *symcl Rel*
∧ (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *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* ⇒ '*b* ⇒ *bool*
shows (∃ *Rel*.
(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *rel-respects-binary-pred Rel Pred*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-respects-binary-pred Rel Pred*)
proof
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*) ∧ *rel-respects-binary-pred Rel Pred*
from *this* **obtain** *Rel*
where ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
and *rel-respects-binary-pred Rel Pred*
by *blast*
thus ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-respects-binary-pred Rel Pred*
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ *rel-respects-binary-pred Rel Pred*
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: *rel-respects-binary-pred Rel Pred*
by *blast*
from *A1* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *symcl Rel*
∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *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* ⇒ *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* ⇒ *bool*
shows *enc-respects-pred Pred*
= (∃ *Rel*.
(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *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* × '*procT*) *set* ⇒ ((('*procS*, '*procT*) *Proc*) × (('*procS*, '*procT*) *Proc*)) *set*
for *TRel* :: ('*procT* × '*procT*) *set*
where
encR: (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelRT TRel* |
target: (*T1*, *T2*) ∈ *TRel* ⟹ (*TargetTerm T1*, *TargetTerm T2*) ∈ *indRelRT TRel*

abbreviation (**in** *encoding*) *indRelRTinfix*
:: ('*procS*, '*procT*) *Proc* ⇒ ('*procT* × '*procT*) *set* ⇒ ('*procS*, '*procT*) *Proc* ⇒ *bool*
(‹- ℛ⟦·⟧*RT*<-> -› [*75*, *75*, *75*] *80*)
where
P ℛ⟦·⟧*RT*<*TRel*> *Q* ≡ (*P*, *Q*) ∈ *indRelRT TRel*

inductive-set (**in** *encoding*) *indRelRTPO*
:: ('*procT* × '*procT*) *set* ⇒ ((('*procS*, '*procT*) *Proc*) × (('*procS*, '*procT*) *Proc*)) *set*
for *TRel* :: ('*procT* × '*procT*) *set*
where
encR: (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelRTPO TRel* |
source: (*SourceTerm S*, *SourceTerm S*) ∈ *indRelRTPO TRel* |
target: (*T1*, *T2*) ∈ *TRel* ⟹ (*TargetTerm T1*, *TargetTerm T2*) ∈ *indRelRTPO TRel* |
trans: ⟦(*P*, *Q*) ∈ *indRelRTPO TRel*; (*Q*, *R*) ∈ *indRelRTPO TRel*⟧ ⟹ (*P*, *R*) ∈ *indRelRTPO TRel*

abbreviation (**in** *encoding*) *indRelRTPOinfix*
:: ('*procS*, '*procT*) *Proc* ⇒ ('*procT* × '*procT*) *set* ⇒ ('*procS*, '*procT*) *Proc* ⇒ *bool*
(‹- ≲⟦·⟧*RT*<-> -› [*75*, *75*, *75*] *80*)
where
P ≲⟦·⟧*RT*<*TRel*> *Q* ≡ (*P*, *Q*) ∈ *indRelRTPO TRel*

lemma (**in** *encoding*) *indRelRTPO-refl*:
fixes *TRel* :: ('*procT* × '*procT*) *set*
assumes *refl*: *refl TRel*
shows *refl* (*indRelRTPO TRel*)
unfolding *refl-on-def*
proof *auto*
fix *P*
show *P* ≲⟦·⟧*RT*<*TRel*> *P*
proof (*cases P*)
case (*SourceTerm SP*)
assume *SP* ∈*S P*
thus *P* ≲⟦·⟧*RT*<*TRel*> *P*
by (*simp add*: *indRelRTPO.source*)
next
case (*TargetTerm TP*)
assume *TP* ∈*T P*
with *refl* **show** *P* ≲⟦·⟧*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* × '*procT*) *set*

assumes *refl*: *refl TRel*
shows *indRelRTPO TRel* = (*indRelRT TRel*)*
proof *auto*
fix *P Q*
assume $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q$
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 $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q$
proof *induct*
from *refl* **show** $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ P$
using *indRelRTPO-refl*[*of TRel*]
unfolding *refl-on-def*
by *simp*
next
case (*step Q R*)
assume $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q$
moreover assume $Q\ \mathcal{R}[\![\cdot]\!]RT{<}TRel{>}\ R$
hence $Q \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R$
by (*induct*, *simp-all add*: *indRelRTPO.encR indRelRTPO.target*)
ultimately show $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R$
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 $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q$ **and** $Q \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R$

thus $P \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R$
using *indRelRTPO.trans*
by *blast*
qed
qed

lemma (**in** *encoding*) *transitive-closure-of-TRel-to-indRelRTPO*:
fixes $TRel$ :: $('procT \times 'procT)\ set$
and $TP\ TQ$ :: $'procT$
shows $(TP,\ TQ) \in TRel^+ \Longrightarrow TargetTerm\ TP \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TQ$
proof −
assume $(TP,\ TQ) \in TRel^+$
thus $TargetTerm\ TP \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TQ$
proof *induct*
fix TQ
assume $(TP,\ TQ) \in TRel$
thus $TargetTerm\ TP \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TQ$
by (*rule indRelRTPO.target*)
next
case (*step TQ TR*)
assume $TargetTerm\ TP \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TQ$
moreover assume $(TQ,\ TR) \in TRel$
hence $TargetTerm\ TQ \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TR$
by (*simp add*: *indRelRTPO.target*)
ultimately show $TargetTerm\ TP \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ TR$
by (*rule indRelRTPO.trans*)
qed
qed

The relation 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 indRelR.

lemma (**in** *encoding*) *indRelR-modulo-pred-impl-indRelRT-modulo-pred*:
fixes $Pred$ :: $(('procS,\ 'procT)\ Proc \times ('procS,\ 'procT)\ Proc) \Rightarrow bool$
shows $(\forall (P,\ Q) \in indRelR.\ Pred\ (P,\ Q)) = (\forall\ TRel.\ (\forall (TP,\ TQ) \in TRel.$
$Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ)) \longleftrightarrow (\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q)))$
proof (*rule iffI*)
assume A: $\forall (P,\ Q) \in indRelR.\ Pred\ (P,\ Q)$
show $\forall\ TRel.\ (\forall (TP,\ TQ) \in TRel.\ Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ))$
$= (\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q))$
proof (*rule allI*, *rule iffI*)
fix $TRel$
assume $\forall (TP,\ TQ) \in TRel.\ Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ)$
with A **show** $\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q)$
by (*auto simp add*: *indRelR.encR indRelRT.simps*)
next
fix $TRel$
assume $\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q)$
thus $\forall (TP,\ TQ) \in TRel.\ Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ)$
by (*auto simp add*: *indRelRT.target*)
qed
next
assume $\forall\ TRel.\ (\forall (TP,\ TQ) \in TRel.\ Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ))$
$\longleftrightarrow (\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q))$
hence B: $\bigwedge TRel.\ (\forall (TP,\ TQ) \in TRel.\ Pred\ (TargetTerm\ TP,\ TargetTerm\ TQ))$
$\longleftrightarrow (\forall (P,\ Q) \in indRelRT\ TRel.\ Pred\ (P,\ Q))$
by *blast*
have $\bigwedge S.\ Pred\ (SourceTerm\ S,\ TargetTerm\ ([\![S]\!]))$
using $B[of\ \{\}]$
by (*simp add*: *indRelRT.simps*)

thus ∀(*P*, *Q*) ∈ *indRelR*. *Pred* (*P*, *Q*)
by (*auto simp add*: *indRelR.simps*)
qed

lemma (**in** *encoding*) *indRelRT-iff-exists-source-target-relation*:
fixes *Pred* :: ((′*procS*, ′*procT*) *Proc* × (′*procS*, ′*procT*) *Proc*) ⇒ *bool*
shows (∀ *TRel*. (∀(*TP*, *TQ*) ∈ *TRel*. *Pred* (*TargetTerm TP*, *TargetTerm TQ*))
⟷ (∀(*P*, *Q*) ∈ *indRelRT TRel*. *Pred* (*P*, *Q*)))
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*) ∧ (∀(*P*, *Q*) ∈ *Rel*. *Pred* (*P*, *Q*)))
using *indRelR-iff-exists-source-target-relation*[**where** *Pred*=*Pred*]
indRelR-modulo-pred-impl-indRelRT-modulo-pred[**where** *Pred*=*Pred*]
by *simp*

lemma (**in** *encoding*) *indRelRT-modulo-pred-impl-indRelRTPO-modulo-pred*:
fixes *TRel* :: (′*procT* × ′*procT*) *set*
and *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*) ∈ *indRelRT TRel*. *Pred* (*P*, *Q*)) = (∀(*P*, *Q*) ∈ *indRelRTPO TRel*. *Pred* (*P*, *Q*))
proof *auto*
fix *P Q*
assume *A*: ∀ *x* ∈ *indRelRT TRel*. *Pred x*
assume *P* ≲⟦·⟧*RT*<*TRel*> *Q*
thus *Pred* (*P*, *Q*)
proof *induct*
case (*encR S*)
have *SourceTerm S* ℛ⟦·⟧*RT*<*TRel*> *TargetTerm* (⟦*S*⟧)
by (*simp add*: *indRelRT.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 T1 T2*)
assume (*T1*, *T2*) ∈ *TRel*
hence *TargetTerm T1* ℛ⟦·⟧*RT*<*TRel*> *TargetTerm T2*
by (*simp add*: *indRelRT.target*)
with *A* **show** *Pred* (*TargetTerm T1*, *TargetTerm T2*)
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* ∈ *indRelRTPO TRel*. *Pred x* **and** *P* ℛ⟦·⟧*RT*<*TRel*> *Q*
thus *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 *Pred* :: ((′*procS*, ′*procT*) *Proc* × (′*procS*, ′*procT*) *Proc*) ⇒ *bool*
assumes ∀ *P*. *Pred* (*P*, *P*)
and ∀ *P Q R*. *Pred* (*P*, *Q*) ∧ *Pred* (*Q*, *R*) ⟶ *Pred* (*P*, *R*)
shows (∀(*P*, *Q*) ∈ *indRelR*. *Pred* (*P*, *Q*))
= (∀ *TRel*. (∀(*TP*, *TQ*) ∈ *TRel*. *Pred* (*TargetTerm TP*, *TargetTerm TQ*))
⟷ (∀(*P*, *Q*) ∈ *indRelRTPO TRel*. *Pred* (*P*, *Q*)))

```
proof −
  have (∀ (P, Q)∈indRelR. Pred (P, Q)) = (∀ TRel. (∀ (TP, TQ) ∈ TRel.
        Pred (TargetTerm TP, TargetTerm TQ)) ⟷ (∀ (P, Q) ∈ indRelRT TRel. Pred (P, Q)))
      using indRelR-modulo-pred-impl-indRelRT-modulo-pred[where Pred=Pred]
    by simp
  moreover
  have ∀ TRel. (∀ (P, Q)∈indRelRT TRel. Pred (P, Q)) = (∀ (P, Q)∈indRelRTPO TRel. 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:   (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRelLT TRel |
  target: (T1, T2) ∈ TRel ⟹ (TargetTerm T1, TargetTerm T2) ∈ indRelLT TRel

abbreviation (in encoding) indRelLTinfix
    :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
      (‹- ℛ⟦·⟧LT<-> -› [75, 75, 75] 80)
  where
  P ℛ⟦·⟧LT<TRel> Q ≡ (P, Q) ∈ indRelLT TRel

inductive-set (in encoding) indRelLTPO
    :: ('procT × 'procT) set ⇒ ((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
    for TRel :: ('procT × 'procT) set
  where
  encL:   (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRelLTPO TRel |
  source: (SourceTerm S, SourceTerm S) ∈ indRelLTPO TRel |
  target: (T1, T2) ∈ TRel ⟹ (TargetTerm T1, TargetTerm T2) ∈ indRelLTPO TRel |
  trans:  ⟦(P, Q) ∈ indRelLTPO TRel; (Q, R) ∈ indRelLTPO TRel⟧ ⟹ (P, R) ∈ indRelLTPO TRel

abbreviation (in encoding) indRelLTPOinfix
    :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
      (‹- ≲⟦·⟧LT<-> -› [75, 75, 75] 80)
  where
  P ≲⟦·⟧LT<TRel> Q ≡ (P, Q) ∈ indRelLTPO TRel

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 ≲⟦·⟧LT<TRel> P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈S P
    thus P ≲⟦·⟧LT<TRel> P
      by (simp add: indRelLTPO.source)
  next
    case (TargetTerm TP)
    assume TP ∈T P
    with refl show P ≲⟦·⟧LT<TRel> 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: $(SourceTerm\ S, TargetTerm\ ([\![S]\!])) \in indRelT\ TRel$ |
encL: $(TargetTerm\ ([\![S]\!]), SourceTerm\ S) \in indRelT\ TRel$ |
target: $(T1, T2) \in TRel \Longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in indRelT\ TRel$

```
abbreviation (in encoding) indRelTinfix
    :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
      (‹- R⟦·⟧T<-> -› [75, 75, 75] 80)
  where
  P R⟦·⟧T<TRel> Q ≡ (P, Q) ∈ indRelT TRel

lemma (in encoding) indRelT-symm:
  fixes TRel :: ('procT × 'procT) set
  assumes symm: sym TRel
  shows sym (indRelT TRel)
    unfolding sym-def
proof clarify
  fix P Q
  assume (P, Q) ∈ indRelT TRel
  thus (Q, P) ∈ 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 × 'procT) set ⇒ ((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
    for TRel :: ('procT × 'procT) set
  where
  encR:   (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelTEQ TRel |
  encL:   (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRelTEQ TRel |
  target: (T1, T2) ∈ TRel ⟹ (TargetTerm T1, TargetTerm T2) ∈ indRelTEQ TRel |
  trans:  ⟦(P, Q) ∈ indRelTEQ TRel; (Q, R) ∈ indRelTEQ TRel⟧ ⟹ (P, R) ∈ indRelTEQ TRel

abbreviation (in encoding) indRelTEQinfix
    :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
      (‹- ∼⟦·⟧T<-> -› [75, 75, 75] 80)
  where
  P ∼⟦·⟧T<TRel> Q ≡ (P, Q) ∈ indRelTEQ TRel

lemma (in encoding) indRelTEQ-refl:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows refl (indRelTEQ TRel)
    unfolding refl-on-def
proof auto
  fix P
  show P ∼⟦·⟧T<TRel> P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈S P
    moreover have SourceTerm SP ∼⟦·⟧T<TRel> TargetTerm (⟦SP⟧)
      by (rule indRelTEQ.encR)
    moreover have TargetTerm (⟦SP⟧) ∼⟦·⟧T<TRel> SourceTerm SP
      by (rule indRelTEQ.encL)
    ultimately show P ∼⟦·⟧T<TRel> P
      by (simp add: indRelTEQ.trans[where P=SourceTerm SP and Q=TargetTerm (⟦SP⟧)])
  next
    case (TargetTerm TP)
    assume TP ∈T P
    with refl show P ∼⟦·⟧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 ∼⟦·⟧T<TRel> Q
  thus Q ∼⟦·⟧T<TRel> P
  proof induct
    case (encR S)
    show TargetTerm (⟦S⟧) ∼⟦·⟧T<TRel> SourceTerm S
      by (rule indRelTEQ.encL)
  next
    case (encL S)
    show SourceTerm S ∼⟦·⟧T<TRel> TargetTerm (⟦S⟧)
      by (rule indRelTEQ.encR)
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    with symm show TargetTerm T2 ∼⟦·⟧T<TRel> TargetTerm T1
        unfolding sym-def
      by (simp add: indRelTEQ.target)
  next
    case (trans P Q R)
    assume R ∼⟦·⟧T<TRel> Q and Q ∼⟦·⟧T<TRel> P
    thus R ∼⟦·⟧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 ∼⟦·⟧T<TRel> Q
  thus (P, Q) ∈ (indRelT TRel)*
  proof induct
    case (encR S)
    show (SourceTerm S, TargetTerm (⟦S⟧)) ∈ (indRelT TRel)*
        using indRelT.encR[of S TRel]
      by simp
  next
    case (encL S)
    show (TargetTerm (⟦S⟧), SourceTerm S) ∈ (indRelT TRel)*
        using indRelT.encL[of S TRel]
      by simp
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    thus (TargetTerm T1, TargetTerm T2) ∈ (indRelT TRel)*
        using indRelT.target[of T1 T2 TRel]
      by simp
  next
    case (trans P Q R)
    assume (P, Q) ∈ (indRelT TRel)* and (Q, R) ∈ (indRelT TRel)*
    thus (P, R) ∈ (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
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⟧) R⟦·⟧T<TRel> SourceTerm S
        by (rule indRelT.encL)
    next
      fix T1 T2
      assume (T1, T2) ∈ TRel
      with symm show TargetTerm T2 R⟦·⟧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 × '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 R⟦·⟧T<TRel> Q
    thus (P, Q) ∈ symcl (indRelLT TRel)
      by (induct, simp-all add: symcl-def indRelLT.encL indRelLT.target)
  next
    fix P Q
    assume (P, Q) ∈ symcl (indRelLT TRel)
    thus P R⟦·⟧T<TRel> Q
    proof (auto simp add: symcl-def indRelLT.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⟧) R⟦·⟧T<TRel> SourceTerm S
        by (rule indRelT.encL)
    next
      fix T1 T2
      assume (T1, T2) ∈ TRel
      with symm show TargetTerm T2 R⟦·⟧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 indRelRT, indRelLT, or 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 $TRel$ :: $('procT \times 'procT)$ *set*
    and TP TQ :: $'procT$
  assumes *rel*: $TargetTerm\ TP\ \mathcal{R}[\![\cdot]\!]RT\text{<}TRel\text{>}\ TargetTerm\ TQ$
  shows $(TP,\ TQ) \in TRel$
      using *rel*
    by (*simp add*: *indRelRT.simps*)

lemma (**in** *encoding*) *indRelLT-to-TRel*:
  fixes $TRel$ :: $('procT \times 'procT)$ *set*
    and TP TQ :: $'procT$
  assumes *rel*: $TargetTerm\ TP\ \mathcal{R}[\![\cdot]\!]LT\text{<}TRel\text{>}\ TargetTerm\ TQ$
  shows $(TP,\ TQ) \in TRel$
      using *rel*
    by (*simp add*: *indRelLT.simps*)

lemma (**in** *encoding*) *indRelT-to-TRel*:
  fixes $TRel$ :: $('procT \times 'procT)$ *set*
    and TP TQ :: $'procT$
  assumes *rel*: $TargetTerm\ TP\ \mathcal{R}[\![\cdot]\!]T\text{<}TRel\text{>}\ TargetTerm\ TQ$
  shows $(TP,\ TQ) \in TRel$
      using *rel*
    by (*simp add*: *indRelT.simps*)

If the preorders indRelRTPO, indRelLTPO, or the equivalence 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 $TRel$ :: $('procT \times 'procT)$ *set*
    and P Q :: $('procS,\ 'procT)$ *Proc*
  assumes *rel*: $P \lesssim[\![\cdot]\!]RT\text{<}TRel\text{>}\ Q$
  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 $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q \longrightarrow False$
    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 $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q \longrightarrow False$
          and $\forall TP\ TQ.\ TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP,\ TQ) \in TRel^+$
  proof *induct*
    case (*encR S*)
    show $\forall SP\ SQ.\ SP \in S\ SourceTerm\ S \wedge SQ \in S\ TargetTerm\ ([\![S]\!]) \longrightarrow SP = SQ$
     and $\forall TP\ SQ.\ TP \in T\ SourceTerm\ S \wedge SQ \in S\ TargetTerm\ ([\![S]\!]) \longrightarrow False$
     and $\forall TP\ TQ.\ TP \in T\ SourceTerm\ S \wedge TQ \in T\ TargetTerm\ ([\![S]\!]) \longrightarrow (TP,\ TQ) \in TRel^+$
      by *simp-all*
    from *reflTRel* **show** $\forall SP\ TQ.\ SP \in S\ SourceTerm\ S \wedge TQ \in T\ TargetTerm\ ([\![S]\!])$
                $\longrightarrow ([\![SP]\!],\ TQ) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
      by *blast*
  next
    case (*source S*)
    show $\forall SP\ SQ.\ SP \in S\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S \longrightarrow SP = SQ$

by *simp*
show $\forall SP\ TQ.\ SP \in S\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S$
$\longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2).\ \exists S.\ T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
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 $(TQ, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
by *blast*
next
case $(step\ T2\ T3)$
assume $(TQ, T2) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
moreover assume $(T2, T3) \in TRel$
hence $(T2, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
by *blast*
ultimately show $(TQ, T3) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
by *simp*
qed
ultimately show $([\![SP]\!], TR) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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*: $P \lesssim[\![\cdot]\!]LT<TRel>\ Q$
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 $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q$
$\longrightarrow (TP, [\![SQ]\!]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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 $\forall SP\ TQ.\ SP \in S\ P \wedge TQ \in T\ Q \longrightarrow False$
and $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q$
$\longrightarrow (TP, [\![SQ]\!]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$

and $\forall TP\ TQ.\ TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP,\ TQ) \in TRel^+$
proof *induct*
case (*encL S*)
show $\forall SP\ SQ.\ SP \in S\ TargetTerm\ ([\![S]\!]) \wedge SQ \in S\ SourceTerm\ S \longrightarrow SP = SQ$
and $\forall SP\ TQ.\ SP \in S\ TargetTerm\ ([\![S]\!]) \wedge TQ \in T\ SourceTerm\ S \longrightarrow False$
and $\forall TP\ TQ.\ TP \in T\ TargetTerm\ ([\![S]\!]) \wedge TQ \in T\ SourceTerm\ S \longrightarrow (TP,\ TQ) \in TRel^+$
by *simp-all*
from *reflTRel* **show** $\forall TP\ SQ.\ TP \in T\ TargetTerm\ ([\![S]\!]) \wedge SQ \in S\ SourceTerm\ S$
$\longrightarrow (TP,\ [\![SQ]\!]) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
by *blast*
next
case (*source S*)
show $\forall SP\ SQ.\ SP \in S\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S \longrightarrow SP = SQ$
by *simp*
show $\forall SP\ TQ.\ SP \in S\ SourceTerm\ S \wedge TQ \in T\ SourceTerm\ S \longrightarrow False$
and $\forall TP\ SQ.\ TP \in T\ SourceTerm\ S \wedge SQ \in S\ SourceTerm\ S$
$\longrightarrow (TP,\ [\![SQ]\!]) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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 $\forall TP\ SQ.\ TP \in T\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2$
$\longrightarrow (TP,\ [\![SQ]\!]) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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 $A3$: $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q$
$\longrightarrow (TP,\ [\![SQ]\!]) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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 $A7$: $\forall TQ\ SR.\ TQ \in T\ Q \wedge SR \in S\ R$
$\longrightarrow (TQ,\ [\![SR]\!]) \in (TRel \cup \{(T1,\ T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
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 $TRel :: ('procT \times 'procT)\ set$

```
    and P Q :: ('procS, 'procT) Proc
  assumes rel: P ∼⟦·⟧T<TRel> Q
  shows ∀SP SQ. SP ∈S P ∧ SQ ∈S Q
         ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
    and ∀SP TQ. SP ∈S P ∧ TQ ∈T Q
         ⟶ (⟦SP⟧, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
    and ∀TP SQ. TP ∈T P ∧ SQ ∈S Q
         ⟶ (TP, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
    and ∀TP TQ. TP ∈T P ∧ TQ ∈T Q
         ⟶ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
proof −
  have reflTRel: ∀S. (⟦S⟧, ⟦S⟧) ∈ TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧}
    by auto
  from rel show ∀SP SQ. SP ∈S P ∧ SQ ∈S Q
                ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
            and ∀SP TQ. SP ∈S P ∧ TQ ∈T Q
                ⟶ (⟦SP⟧, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
            and ∀TP SQ. TP ∈T P ∧ SQ ∈S Q
                ⟶ (TP, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
            and ∀TP TQ. TP ∈T P ∧ TQ ∈T Q
                ⟶ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
  proof induct
    case (encR S)
    show ∀SP SQ. SP ∈S SourceTerm S ∧ SQ ∈S TargetTerm (⟦S⟧)
          ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀TP SQ. TP ∈T SourceTerm S ∧ SQ ∈S TargetTerm (⟦S⟧)
          ⟶ (TP, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀TP TQ. TP ∈T SourceTerm S ∧ TQ ∈T TargetTerm (⟦S⟧)
          ⟶ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by simp+
    from reflTRel show ∀SP TQ. SP ∈S SourceTerm S ∧ TQ ∈T TargetTerm (⟦S⟧)
                       ⟶ (⟦SP⟧, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by blast
  next
    case (encL S)
    show ∀SP SQ. SP ∈S TargetTerm (⟦S⟧) ∧ SQ ∈S SourceTerm S
          ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀SP TQ. SP ∈S TargetTerm (⟦S⟧) ∧ TQ ∈T SourceTerm S
          ⟶ (⟦SP⟧, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀TP TQ. TP ∈T TargetTerm (⟦S⟧) ∧ TQ ∈T SourceTerm S
          ⟶ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by simp+
    from reflTRel show ∀TP SQ. TP ∈T TargetTerm (⟦S⟧) ∧ SQ ∈S SourceTerm S
                       ⟶ (TP, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by blast
  next
    case (target T1 T2)
    show ∀SP SQ. SP ∈S TargetTerm T1 ∧ SQ ∈S TargetTerm T2
          ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀SP TQ. SP ∈S TargetTerm T1 ∧ TQ ∈T TargetTerm T2
          ⟶ (⟦SP⟧, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
     and ∀TP SQ. TP ∈T TargetTerm T1 ∧ SQ ∈S TargetTerm T2
          ⟶ (TP, ⟦SQ⟧) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by simp+
    assume (T1, T2) ∈ TRel
    thus ∀TP TQ. TP ∈T TargetTerm T1 ∧ TQ ∈T TargetTerm T2
          ⟶ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = ⟦S⟧ ∧ T2 = ⟦S⟧})^+
      by blast
  next
    case (trans P Q R)
    assume A1: ∀SP SQ. SP ∈S P ∧ SQ ∈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 $TQ \in T\ Q$
    with *A4 A8* **show** *?thesis*
      by *auto*
  qed
 qed
qed

lemma (**in** *encoding*) *trans-closure-of-TRel-refl-cond*:
 fixes $TRel$ :: $('procT \times 'procT)$ *set*
  and $TP\ TQ$ :: $'procT$
 assumes $(TP, TQ) \in (TRel \cup \{(T1, T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\})^+$
 shows $(TP, TQ) \in TRel^*$
  using *assms*
proof *induct*
 fix TQ
 assume $(TP, TQ) \in TRel \cup \{(T1, T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\}$
 thus $(TP, TQ) \in TRel^*$
  by *auto*
next
 case (*step TQ TR*)
 assume $(TP, TQ) \in TRel^*$
 moreover assume $(TQ, TR) \in TRel \cup \{(T1, T2).\ \exists S.\ T1 = [\![S]\!] \wedge T2 = [\![S]\!]\}$
 hence $(TQ, TR) \in TRel^*$
  by *blast*
 ultimately show $(TP, TR) \in TRel^*$
  by *simp*
qed

Note that if indRelRTPO relates a source term S to a target term T, then the translation of S is equal to T or indRelRTPO also relates the translation of S to T.

lemma (**in** *encoding*) *indRelRTPO-relates-source-target*:
 fixes $TRel$ :: $('procT \times 'procT)$ *set*
  and S :: $'procS$
  and T :: $'procT$
 assumes *pair*: $SourceTerm\ S \lesssim[\![\cdot]\!]RT<TRel>\ TargetTerm\ T$
 shows $(TargetTerm\ ([\![S]\!]), TargetTerm\ T) \in (indRelRTPO\ TRel)^=$
proof −
 from *pair* **have** $([\![S]\!], T) \in TRel^*$
   using *indRelRTPO-to-TRel*(*2*)[**where** *TRel*=*TRel*] *trans-closure-of-TRel-refl-cond*
  by *simp*
 hence $[\![S]\!] = T \vee ([\![S]\!], T) \in TRel^+$
   using *rtrancl-eq-or-trancl*[*of* $[\![S]\!]$ *T TRel*]
  by *blast*
 moreover have $[\![S]\!] = T \Longrightarrow (TargetTerm\ ([\![S]\!]), TargetTerm\ T) \in (indRelRTPO\ TRel)^=$
  by *simp*
 moreover
 have $([\![S]\!], T) \in TRel^+ \Longrightarrow (TargetTerm\ ([\![S]\!]), TargetTerm\ T) \in (indRelRTPO\ TRel)^=$
   using *transitive-closure-of-TRel-to-indRelRTPO*[**where** *TRel*=*TRel*]
  by *simp*
 ultimately show $(TargetTerm\ ([\![S]\!]), TargetTerm\ T) \in (indRelRTPO\ TRel)^=$
  by *blast*
qed

If indRelRTPO, indRelLTPO, or indRelTPO preserves barbs then so does the corresponding target term relation.

lemma (**in** *encoding-wrt-barbs*) *rel-with-target-impl-TRel-preserves-barbs*:
 fixes $TRel$ :: $('procT \times 'procT)$ *set*
  and Rel :: $(('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ *set*
 assumes *preservation*: *rel-preserves-barbs Rel* (*STCalWB SWB TWB*)

and *targetInRel*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
shows *rel-preserves-barbs TRel TWB*
proof *clarify*
fix *TP TQ a*
assume (*TP*, *TQ*) ∈ *TRel*
with *targetInRel* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *blast*
moreover assume *TP*↓<*TWB*>*a*
hence *TargetTerm TP*↓.*a*
by *simp*
ultimately have *TargetTerm TQ*↓.*a*
using *preservation preservation-of-barbs-in-barbed-encoding*[**where** *Rel*=*Rel*]
by *blast*
thus *TQ*↓<*TWB*>*a*
by *simp*
qed

lemma (**in** *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-preserves-barbs*:
fixes *TRel* :: (′*procT* × ′*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* × ′*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* × ′*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* × ′*procT*) *set*
and *Rel* :: ((′*procS*, ′*procT*) *Proc* × (′*procS*, ′*procT*) *Proc*) *set*
assumes *preservation*: *rel-weakly-preserves-barbs Rel* (*STCalWB SWB TWB*)
and *targetInRel*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
shows *rel-weakly-preserves-barbs TRel TWB*
proof *clarify*
fix *TP TQ a TP*′
assume (*TP*, *TQ*) ∈ *TRel*
with *targetInRel* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *blast*
moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP*′ **and** *TP*′↓<*TWB*>*a*
hence *TargetTerm TP*⇓.*a*
by *blast*
ultimately have *TargetTerm TQ*⇓.*a*
using *preservation weak-preservation-of-barbs-in-barbed-encoding*[**where** *Rel*=*Rel*]
by *blast*
thus *TQ*⇓<*TWB*>*a*
by *simp*
qed

lemma (**in** *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-preserves-barbs*:
fixes *TRel* :: (*'procT* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *set*
and *Rel* :: ((*'procS*, *'procT*) *Proc* × (*'procS*, *'procT*) *Proc*) *set*
assumes *reflection*: *rel-reflects-barbs Rel* (*STCalWB SWB TWB*)
and *targetInRel*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
shows *rel-reflects-barbs TRel TWB*
proof *clarify*
fix *TP TQ a*
assume (*TP*, *TQ*) ∈ *TRel*
with *targetInRel* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *blast*
moreover assume *TQ*↓<*TWB*>*a*
hence *TargetTerm TQ*↓.*a*
by *simp*
ultimately have *TargetTerm TP*↓.*a*
using *reflection 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-reflects-barbs*:
fixes *TRel* :: (*'procT* × *'procT*) *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* × *'procT*) *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* :: ((*'procS*, *'procT*) *Proc* × (*'procS*, *'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* ⟼(*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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *set*
and *Rel* :: ((*'procS*, *'procT*) *Proc* × (*'procS*, *'procT*) *Proc*) *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* $\longmapsto$*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 $TP \longmapsto Target{*}\ TP'$
hence $TargetTerm\ TP \longmapsto (STCal\ Source\ Target){*}\ (TargetTerm\ TP')$
by (*simp add*: *STCal-steps*)
ultimately obtain Q' **where** *A2*: $TargetTerm\ TQ \longmapsto (STCal\ Source\ Target){*}\ Q'$
and *A3*: $(TargetTerm\ TP', Q') \in Rel$
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*)
from *A3 A5 trel* **have** $(TP', TQ') \in TRel^+$
by *simp*
with *A4* **show** $\exists\, TQ'.\ TQ \longmapsto Target{*}\ TQ' \wedge (TP', TQ') \in TRel^+$
by *blast*
next
case (*step TQ TR*)
assume $TP \longmapsto Target{*}\ TP'$
and $\bigwedge TP'.\ TP \longmapsto Target{*}\ TP' \Longrightarrow \exists\, TQ'.\ TQ \longmapsto Target{*}\ TQ' \wedge (TP', TQ') \in TRel^+$
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-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-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*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
assumes *sim*: *weak-reduction-simulation* (*indRelRTPO TRel*) (*STCal Source Target*)
shows *weak-reduction-simulation* ($TRel^+$) *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* :: $('procT \times 'procT)$ *set*
assumes *sim*: *weak-reduction-simulation* (*indRelLTPO TRel*) (*STCal Source Target*)
shows *weak-reduction-simulation* ($TRel^+$) *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* × *'procT*) *set*
    and *Rel* :: ((*'procS*, *'procT*) *Proc* × (*'procS*, *'procT*) *Proc*) *set*
  assumes *sim*: *weak-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
      and *target*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
      and *trel*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^{+}$
  shows *weak-reduction-simulation* (($TRel^{+}$)$^{-1}$) *Target*
proof *clarify*
  fix *TP TQ TP'*
  assume (*TQ*, *TP*) ∈ $TRel^{+}$
  moreover assume *TP* ⟼*Target*∗ *TP'*
  ultimately show ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TP'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
  proof (*induct arbitrary*: *TP'*)
    fix *TP TP'*
    assume (*TQ*, *TP*) ∈ *TRel*
    with *target* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ Rel^{-1}
      by *simp*
    moreover assume *TP* ⟼*Target*∗ *TP'*
    hence *TargetTerm TP* ⟼(*STCal Source Target*)∗ (*TargetTerm TP'*)
      by (*simp add*: *STCal-steps*)
    ultimately obtain *Q'* **where** *A2*: *TargetTerm TQ* ⟼(*STCal Source Target*)∗ *Q'*
                      and *A3*: (*TargetTerm TP'*, *Q'*) ∈ Rel^{-1}
        using *sim*
      by *blast*
    from *A2* **obtain** *TQ'* **where** *A4*: *TQ* ⟼*Target*∗ *TQ'* **and** *A5*: *TQ'* ∈*T* *Q'*
      by (*auto simp add*: *STCal-steps*(*2*))
    from *A3 A5 trel* **have** (*TP'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
      by *simp*
    with *A4* **show** ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TP'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
      by *blast*
  next
    case (*step TR TP TP'*)
    assume *TP* ⟼*Target*∗ *TP'*
    hence *TargetTerm TP* ⟼(*STCal Source Target*)∗ (*TargetTerm TP'*)
      by (*simp add*: *STCal-steps*)
    moreover assume (*TR*, *TP*) ∈ *TRel*
    with *target* **have** (*TargetTerm TP*, *TargetTerm TR*) ∈ Rel^{-1}
      by *simp*
    ultimately obtain *R'* **where** *B1*: *TargetTerm TR* ⟼(*STCal Source Target*)∗ *R'*
                      and *B2*: (*TargetTerm TP'*, *R'*) ∈ Rel^{-1}
        using *sim*
      by *blast*
    from *B1* **obtain** *TR'* **where** *B3*: *TR'* ∈*T* *R'* **and** *B4*: *TR* ⟼*Target*∗ *TR'*
      by (*auto simp add*: *STCal-steps*)
    assume ⋀*TR'*. *TR* ⟼*Target*∗ *TR'* ⟹ ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TR'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
    with *B4* **obtain** *TQ'* **where** *B5*: *TQ* ⟼*Target*∗ *TQ'* **and** *B6*: (*TR'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
      by *blast*
    from *B6* **have** (*TQ'*, *TR'*) ∈ $TRel^{+}$
      by *simp*
    moreover from *B2 B3 trel* **have** (*TR'*, *TP'*) ∈ $TRel^{+}$
      by *simp*
    ultimately have (*TP'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
      by *simp*
    with *B5* **show** ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TP'*, *TQ'*) ∈ ($TRel^{+}$)$^{-1}$
      by *blast*
  qed
qed

lemma (**in** *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev*:
  fixes *TRel* :: (*'procT* × *'procT*) *set*

assumes *sim*: *weak-reduction-simulation* $((indRelRTPO\ TRel)^{-1})$ (*STCal Source Target*)
shows *weak-reduction-simulation* $((TRel^{+})^{-1})$ *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-rev[**where**
Rel=*indRelRTPO TRel* **and** *TRel*=*TRel*]
by *blast*

lemma (**in** *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
assumes *sim*: *weak-reduction-simulation* $((indRelLTPO\ TRel)^{-1})$ (*STCal Source Target*)
shows *weak-reduction-simulation* $((TRel^{+})^{-1})$ *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-rev[**where**
Rel=*indRelLTPO TRel* **and** *TRel*=*TRel*]
by *blast*

lemma (**in** *encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
and *Rel* :: $(('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ *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 $\exists\ TQ'.\ TQ \longmapsto Target{*}\ TQ' \wedge (TP', TQ') \in TRel^{*}$
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 $\bigwedge TP'.\ TP \longmapsto Target{*}\ TP' \Longrightarrow \exists\ TQ'.\ TQ \longmapsto Target{*}\ TQ' \wedge (TP', TQ') \in TRel^{*}$
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-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-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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *set*
 and *Rel* :: (($'procS$, $'procT$) *Proc* $\times$ ($'procS$, $'procT$) *Proc*) *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* $\longmapsto$*Target TP'*
thus $\exists$ *TQ'*. *TQ* $\longmapsto$*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* $\longmapsto$*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*
   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*)
 from *A3 A5 trel* **have** (*TP'*, *TQ'*) $\in$ $TRel^+$
  by *simp*
 with *A4* **show** $\exists$ *TQ'*. *TQ* $\longmapsto$*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'* $\Longrightarrow$ $\exists$ *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-step*)
 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-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* $\longmapsto$*Target TR'* $\wedge$ (*TP'*, *TR'*) $\in$ $TRel^+$
  by *blast*

qed
qed

lemma (**in** *encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-simulation*:
fixes *TRel* :: ($'procT$ × $'procT$) *set*
assumes *sim*: *strong-reduction-simulation* (*indRelRTPO TRel*) (*STCal Source Target*)
shows *strong-reduction-simulation* ($TRel^+$) *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* :: ($'procT$ × $'procT$) *set*
assumes *sim*: *strong-reduction-simulation* (*indRelLTPO TRel*) (*STCal Source Target*)
shows *strong-reduction-simulation* ($TRel^+$) *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* :: ($'procT$ × $'procT$) *set*
and *Rel* :: (($'procS$, $'procT$) *Proc* × ($'procS$, $'procT$) *Proc*) *set*
assumes *sim*: *strong-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
and *target*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and *trel*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$
shows *strong-reduction-simulation* ($(TRel^+)^{-1}$) *Target*
proof *clarify*
fix *TP TQ TP'*
assume (*TQ*, *TP*) ∈ $TRel^+$
moreover assume *TP* ⟼*Target TP'*
ultimately show ∃ *TQ'*. *TQ* ⟼*Target TQ'* ∧ (*TP'*, *TQ'*) ∈ $(TRel^+)^{-1}$
proof (*induct arbitrary*: *TP'*)
fix *TP TP'*
assume (*TQ*, *TP*) ∈ *TRel*
with *target* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ Rel^{-1}
by *simp*
moreover assume *TP* ⟼*Target TP'*
hence *TargetTerm TP* ⟼(*STCal Source Target*) (*TargetTerm TP'*)
by (*simp add*: *STCal-step*)
ultimately obtain *Q'* **where** *A2*: *TargetTerm TQ* ⟼(*STCal Source Target*) *Q'*
and *A3*: (*TargetTerm TP'*, *Q'*) ∈ Rel^{-1}
using *sim*
by *blast*
from *A2* **obtain** *TQ'* **where** *A4*: *TQ* ⟼*Target TQ'* **and** *A5*: *TQ'* ∈*T Q'*
by (*auto simp add*: *STCal-step*(*2*))
from *A3 A5 trel* **have** (*TP'*, *TQ'*) ∈ $(TRel^+)^{-1}$
by *simp*
with *A4* **show** ∃ *TQ'*. *TQ* ⟼*Target TQ'* ∧ (*TP'*, *TQ'*) ∈ $(TRel^+)^{-1}$
by *blast*
next
case (*step TP TR TR'*)
assume (*TP*, *TR*) ∈ *TRel*
with *target* **have** (*TargetTerm TP*, *TargetTerm TR*) ∈ *Rel*
by *simp*
moreover assume *TR* ⟼*Target TR'*
hence *TargetTerm TR* ⟼(*STCal Source Target*) (*TargetTerm TR'*)
by (*simp add*: *STCal-step*)
ultimately obtain *P'* **where** *B1*: *TargetTerm TP* ⟼(*STCal Source Target*) *P'*
and *B2*: (*P'*, *TargetTerm TR'*) ∈ *Rel*

using *sim*
by *blast*
from *B1* **obtain** TP' **where** *B3*: $TP' \in T\ P'$ **and** *B4*: $TP \longmapsto Target\ TP'$
by (*auto simp add*: *STCal-step*)
assume $\bigwedge TP'.\ TP \longmapsto Target\ TP' \Longrightarrow \exists\, TQ'.\ TQ \longmapsto Target\ TQ' \wedge (TP',\ TQ') \in (TRel^+)^{-1}$
with *B4* **obtain** TQ' **where** *B5*: $TQ \longmapsto Target\ TQ'$ **and** *B6*: $(TP',\ TQ') \in (TRel^+)^{-1}$
by *blast*
from *B2 B3 trel* **have** $(TP',\ TR') \in TRel^+$
by *simp*
with *B6* **have** $(TR',\ TQ') \in (TRel^+)^{-1}$
by *simp*
with *B5* **show** $\exists\, TQ'.\ TQ \longmapsto Target\ TQ' \wedge (TR',\ TQ') \in (TRel^+)^{-1}$
by *blast*
qed
qed

lemma (**in** *encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
assumes *sim*: *strong-reduction-simulation* $((indRelRTPO\ TRel)^{-1})$ (*STCal Source Target*)
shows *strong-reduction-simulation* $((TRel^+)^{-1})$ *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-rev[**where**
Rel=indRelRTPO TRel **and** *TRel=TRel*]
by *blast*

lemma (**in** *encoding*) *indRelLTPO-impl-TRel-is-strong-reduction-simulation-rev*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
assumes *sim*: *strong-reduction-simulation* $((indRelLTPO\ TRel)^{-1})$ (*STCal Source Target*)
shows *strong-reduction-simulation* $((TRel^+)^{-1})$ *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-rev[**where**
Rel=indRelLTPO TRel **and** *TRel=TRel*]
by *blast*

lemma (**in** *encoding*) *rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation*:
fixes *TRel* :: $('procT \times 'procT)$ *set*
and *Rel* :: $(('procS,\ 'procT)\ Proc \times ('procS,\ 'procT)\ Proc)$ *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** $TP \longmapsto Target\ TP'$
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* $TQ\ TR\ TP'$)
assume $TP \longmapsto Target\ TP'$
and $\bigwedge TP'.\ TP \longmapsto Target\ TP' \Longrightarrow \exists\, TQ'.\ TQ \longmapsto Target\ TQ' \wedge (TP',\ TQ') \in TRel^*$
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-step*)
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-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* $\longmapsto$*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*$^+$) *TWB*
proof
from *sim* **show** *weak-reduction-simulation* (*TRel*$^+$) (*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*$^+$) *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*$^+$) *TWB*
proof
from *sim* **show** *weak-reduction-simulation* (*TRel*$^+$) (*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*$^+$) *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* $(TRel^+)$ *TWB*
proof
from *sim* **show** *strong-reduction-simulation* $(TRel^+)$ (*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* $(TRel^+)$ *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* $(TRel^+)$ *TWB*
proof
from *sim refl* **show** *strong-reduction-simulation* $(TRel^+)$ (*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* $(TRel^+)$ *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* × *'procT*) *set*
and *Rel* :: ((*'procS*, *'procT*) *Proc* × (*'procS*, *'procT*) *Proc*) *set*
assumes *conSim*: *weak-reduction-contrasimulation Rel* (*STCal Source Target*)
and *target*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and *trel*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*$^+$
shows *weak-reduction-contrasimulation* (*TRel*$^+$) *Target*
proof *clarify*
fix *TP TQ TP′*
assume (*TP*, *TQ*) ∈ *TRel*$^+$ **and** *TP* ⟼*Target*∗ *TP′*
thus ∃ *TQ′*. *TQ* ⟼*Target*∗ *TQ′* ∧ (*TQ′*, *TP′*) ∈ *TRel*$^+$
proof (*induct arbitrary*: *TP′*)
fix *TQ TP′*
assume (*TP*, *TQ*) ∈ *TRel*
with *target* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP* ⟼*Target*∗ *TP′*
hence *TargetTerm TP* ⟼(*STCal Source Target*)∗ (*TargetTerm TP′*)
by (*simp add*: *STCal-steps*)
ultimately obtain *Q′* **where** *A2*: *TargetTerm TQ* ⟼(*STCal Source Target*)∗ *Q′*
and *A3*: (*Q′*, *TargetTerm TP′*) ∈ *Rel*
using *conSim*
by *blast*
from *A2* **obtain** *TQ′* **where** *A4*: *TQ* ⟼*Target*∗ *TQ′* **and** *A5*: *TQ′* ∈*T* *Q′*
by (*auto simp add*: *STCal-steps*)
from *A3 A5 trel* **have** (*TQ′*, *TP′*) ∈ *TRel*$^+$
by *simp*
with *A4* **show** ∃ *TQ′*. *TQ* ⟼*Target*∗ *TQ′* ∧ (*TQ′*, *TP′*) ∈ *TRel*$^+$
by *blast*
next
case (*step TQ TR*)
assume *TP* ⟼*Target*∗ *TP′*
and ⋀*TP′*. *TP* ⟼*Target*∗ *TP′* ⟹ ∃ *TQ′*. *TQ* ⟼*Target*∗ *TQ′* ∧ (*TQ′*, *TP′*) ∈ *TRel*$^+$
from *this* **obtain** *TQ′* **where** *B1*: *TQ* ⟼*Target*∗ *TQ′* **and** *B2*: (*TQ′*, *TP′*) ∈ *TRel*$^+$
by *blast*
assume (*TQ*, *TR*) ∈ *TRel*
with *target* **have** (*TargetTerm TQ*, *TargetTerm TR*) ∈ *Rel*
by *simp*
moreover from *B1* **have** *TargetTerm TQ* ⟼(*STCal Source Target*)∗ (*TargetTerm TQ′*)
by (*simp add*: *STCal-steps*)
ultimately obtain *R′* **where** *B3*: *TargetTerm TR* ⟼(*STCal Source Target*)∗ *R′*
and *B4*: (*R′*, *TargetTerm TQ′*) ∈ *Rel*
using *conSim*
by *blast*
from *B3* **obtain** *TR′* **where** *B5*: *TR′* ∈*T* *R′* **and** *B6*: *TR* ⟼*Target*∗ *TR′*
by (*auto simp add*: *STCal-steps*)
from *B4 B5 trel* **have** (*TR′*, *TQ′*) ∈ *TRel*$^+$
by *simp*
from *this B2* **have** (*TR′*, *TP′*) ∈ *TRel*$^+$
by *simp*
with *B6* **show** ∃ *TR′*. *TR* ⟼*Target*∗ *TR′* ∧ (*TR′*, *TP′*) ∈ *TRel*$^+$
by *blast*
qed
qed

lemma (**in** *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation*:
fixes *TRel* :: (*'procT* × *'procT*) *set*
assumes *conSim*: *weak-reduction-contrasimulation* (*indRelRTPO TRel*) (*STCal Source Target*)
shows *weak-reduction-contrasimulation* (*TRel*$^+$) *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* **and** *TRel*=*TRel*]

by *blast*

lemma (**in** *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation*:
fixes *TRel* :: ('*procT* × '*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 **and** *TRel=TRel*]
by *blast*

lemma (**in** *encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-contrasimulation*:
fixes *TRel* :: ('*procT* × '*procT*) *set*
and *Rel* :: (('*procS*, '*procT*) *Proc* × ('*procS*, '*procT*) *Proc*) *set*
assumes *conSim*: *weak-reduction-contrasimulation Rel* (*STCal Source Target*)
and *target*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and *trel*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^*$
shows *weak-reduction-contrasimulation* ($TRel^*$) *Target*
proof *clarify*
fix *TP TQ TP'*
assume (*TP*, *TQ*) ∈ $TRel^*$ **and** *TP* ⟼*Target*∗ *TP'*
thus ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TQ'*, *TP'*) ∈ $TRel^*$
proof (*induct arbitrary*: *TP'*)
fix *TP'*
assume *TP* ⟼*Target*∗ *TP'*
moreover have (*TP'*, *TP'*) ∈ $TRel^*$
by *simp*
ultimately show ∃ *TQ'*. *TP* ⟼*Target*∗ *TQ'* ∧ (*TQ'*, *TP'*) ∈ $TRel^*$
by *blast*
next
case (*step TQ TR*)
assume *TP* ⟼*Target*∗ *TP'*
and ⋀*TP'*. *TP* ⟼*Target*∗ *TP'* ⟹ ∃ *TQ'*. *TQ* ⟼*Target*∗ *TQ'* ∧ (*TQ'*, *TP'*) ∈ $TRel^*$
from *this* **obtain** *TQ'* **where** *B1*: *TQ* ⟼*Target*∗ *TQ'* **and** *B2*: (*TQ'*, *TP'*) ∈ $TRel^*$
by *blast*
assume (*TQ*, *TR*) ∈ *TRel*
with *target* **have** (*TargetTerm TQ*, *TargetTerm TR*) ∈ *Rel*
by *simp*
moreover from *B1* **have** *TargetTerm TQ* ⟼(*STCal Source Target*)∗ (*TargetTerm TQ'*)
by (*simp add*: *STCal-steps*)
ultimately obtain *R'* **where** *B3*: *TargetTerm TR* ⟼(*STCal Source Target*)∗ *R'*
and *B4*: (*R'*, *TargetTerm TQ'*) ∈ *Rel*
using *conSim*
by *blast*
from *B3* **obtain** *TR'* **where** *B5*: *TR'* ∈T *R'* **and** *B6*: *TR* ⟼*Target*∗ *TR'*
by (*auto simp add*: *STCal-steps*)
from *B4 B5 trel* **have** (*TR'*, *TQ'*) ∈ $TRel^*$
by *simp*
from *this B2* **have** (*TR'*, *TP'*) ∈ $TRel^*$
by *simp*
with *B6* **show** ∃ *TR'*. *TR* ⟼*Target*∗ *TR'* ∧ (*TR'*, *TP'*) ∈ $TRel^*$
by *blast*
qed
qed

lemma (**in** *encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation*:
fixes *TRel* :: ('*procT* × '*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* :: ($'procT \times 'procT$) *set*
assumes *conSim*: *weak-barbed-contrasimulation* (*indRelRTPO TRel*) (*STCalWB SWB TWB*)
shows *weak-barbed-contrasimulation* ($TRel^+$) *TWB*
proof
from *conSim* **show** *weak-reduction-contrasimulation* ($TRel^+$) (*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* ($TRel^+$) *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* :: ($'procT \times 'procT$) *set*
assumes *conSim*: *weak-barbed-contrasimulation* (*indRelLTPO TRel*) (*STCalWB SWB TWB*)
shows *weak-barbed-contrasimulation* ($TRel^+$) *TWB*
proof
from *conSim* **show** *weak-reduction-contrasimulation* ($TRel^+$) (*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* ($TRel^+$) *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* :: ($'procT \times 'procT$) *set*
assumes *conSim*: *weak-barbed-contrasimulation* (*indRelTEQ TRel*) (*STCalWB SWB TWB*)
shows *weak-barbed-contrasimulation* ($TRel^*$) *TWB*
proof
from *conSim* **show** *weak-reduction-contrasimulation* ($TRel^*$) (*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* ($TRel^*$) *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* :: ($'procT \times 'procT$) *set*
assumes *couSim*: *weak-reduction-coupled-simulation* (*indRelRTPO TRel*) (*STCal Source Target*)
shows *weak-reduction-coupled-simulation* ($TRel^+$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *set*
and *Rel* :: (($'procS$, $'procT$) *Proc* $\times$ ($'procS$, $'procT$) *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-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 $TQ\ TQ'$
assume $(TP, TQ) \in TRel$
with *target* **have** $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$
by *blast*
moreover assume $TQ \longmapsto Target{*}\ TQ'$
hence $TargetTerm\ TQ \longmapsto (STCal\ Source\ Target){*}\ (TargetTerm\ TQ')$
by (*simp add*: *STCal-steps*)
ultimately obtain $P''\ Q''$ **where** *A2*: $TargetTerm\ TP \longmapsto (STCal\ Source\ Target){*}\ P''$
and *A3*: $TargetTerm\ TQ' \longmapsto (STCal\ Source\ Target){*}\ Q''$ **and** *A4*: $(P'', Q'') \in Rel$
using *corSim*
by *blast*
from *A2* **obtain** TP'' **where** *A5*: $TP \longmapsto Target{*}\ TP''$ **and** *A6*: $TP'' \in T\ P''$
by (*auto simp add*: *STCal-steps*)
from *A3* **obtain** TQ'' **where** *A7*: $TQ' \longmapsto Target{*}\ TQ''$ **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 $\exists TP''\ TQ''.\ TP \longmapsto Target{*}\ TP'' \wedge TQ' \longmapsto Target{*}\ TQ'' \wedge (TP'', TQ'') \in TRel^+$
by *blast*
next
case (*step TQ TR TR'*)
assume $\bigwedge TQ'.\ TQ \longmapsto Target{*}\ TQ' \Longrightarrow \exists TP''\ TQ''.\ TP \longmapsto Target{*}\ TP'' \wedge TQ' \longmapsto Target{*}\ TQ''$
$\wedge (TP'', TQ'') \in TRel^+$
moreover assume $(TQ, TR) \in TRel$
hence $\bigwedge TR'.\ TR \longmapsto Target{*}\ TR'$
$\longrightarrow (\exists TQ''\ TR''.\ TQ \longmapsto Target{*}\ TQ'' \wedge TR' \longmapsto Target{*}\ TR'' \wedge (TQ'', TR'') \in TRel^+)$
proof *clarify*
fix TR'
assume $(TQ, TR) \in TRel$
with *target* **have** $(TargetTerm\ TQ, TargetTerm\ TR) \in Rel$
by *simp*
moreover assume $TR \longmapsto Target{*}\ TR'$
hence $TargetTerm\ TR \longmapsto (STCal\ Source\ Target){*}\ (TargetTerm\ TR')$
by (*simp add*: *STCal-steps*)
ultimately obtain $Q''\ R''$ **where** *B1*: $TargetTerm\ TQ \longmapsto (STCal\ Source\ Target){*}\ Q''$
and *B2*: $TargetTerm\ TR' \longmapsto (STCal\ Source\ Target){*}\ R''$ **and** *B3*: $(Q'', R'') \in Rel$
using *corSim*
by *blast*
from *B1* **obtain** TQ'' **where** *B4*: $TQ'' \in T\ Q''$ **and** *B5*: $TQ \longmapsto Target{*}\ TQ''$
by (*auto simp add*: *STCal-steps*)
from *B2* **obtain** TR'' **where** *B6*: $TR'' \in T\ R''$ **and** *B7*: $TR' \longmapsto Target{*}\ TR''$
by (*auto simp add*: *STCal-steps*)
from *B3 B4 B6 trel* **have** $(TQ'', TR'') \in TRel^+$
by *simp*
with *B5 B7*
show $\exists TQ''\ TR''.\ TQ \longmapsto Target{*}\ TQ'' \wedge TR' \longmapsto Target{*}\ TR'' \wedge (TQ'', TR'') \in TRel^+$
by *blast*
qed
moreover have *trans* $(TRel^+)$
by *simp*
moreover assume $TR \longmapsto Target{*}\ TR'$
ultimately
show $\exists TP''\ TR''.\ TP \longmapsto Target{*}\ TP'' \wedge TR' \longmapsto Target{*}\ TR'' \wedge (TP'', TR'') \in TRel^+$
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* ($TRel^+$) *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* ($TRel^+$) *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'* $\Longrightarrow$ $\exists$ *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'* $\Longrightarrow$ $\exists$ *TQ'' TR''*. *TQ* $\longmapsto$*Target*$*$ *TQ''*
$\wedge$ *TR'* $\longmapsto$*Target*$*$ *TR''* $\wedge$ (*TQ''*, *TR''*) $\in$ $TRel^*$

```
    proof clarify
      fix TR′
      assume (TQ, TR) ∈ TRel
      with target have (TargetTerm TQ, TargetTerm TR) ∈ Rel
        by simp
      moreover assume TR ⟼Target∗ TR′
      hence TargetTerm TR ⟼(STCal Source Target)∗ (TargetTerm TR′)
        by (simp add: STCal-steps)
      ultimately obtain Q″ R″ where B1: TargetTerm TQ ⟼(STCal Source Target)∗ Q″
       and B2: TargetTerm TR′ ⟼(STCal Source Target)∗ R″ and B3: (Q″, R″) ∈ Rel
          using corSim
        by blast
      from B1 obtain TQ″ where B4: TQ″ ∈T Q″ and B5: TQ ⟼Target∗ TQ″
        by (auto simp add: STCal-steps)
      from B2 obtain TR″ where B6: TR″ ∈T R″ and B7: TR′ ⟼Target∗ TR″
        by (auto simp add: STCal-steps)
      from B3 B4 B6 trel have (TQ″, TR″) ∈ TRel*
        by simp
      with B5 B7
      show ∃ TQ″ TR″. TQ ⟼Target∗ TQ″ ∧ TR′ ⟼Target∗ TR″ ∧ (TQ″, TR″) ∈ TRel*
        by blast
    qed
    moreover assume TR ⟼Target∗ TR′
    moreover have trans (TRel*)
        using trans-rtrancl[of TRel]
      by simp
    ultimately show ∃ TP″ TR″. TP ⟼Target∗ TP″ ∧ TR′ ⟼Target∗ TR″ ∧ (TP″, TR″) ∈ 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) indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation:
 fixes TRel :: ('procT × 'procT) set
 assumes corSim: weak-reduction-correspondence-simulation (indRelTEQ TRel) (STCal Source Target)
 shows weak-reduction-correspondence-simulation (TRel*) Target
    using corSim 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-correspondence-simulation[
            where Rel=indRelTEQ TRel and TRel=TRel]
  by blast

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-correspondence-simulation:
 fixes TRel :: ('procT × 'procT) set
 assumes corSim: weak-barbed-correspondence-simulation (indRelRTPO TRel) (STCalWB SWB TWB)
 shows weak-barbed-correspondence-simulation (TRel⁺) TWB
proof
 from corSim show weak-reduction-correspondence-simulation (TRel⁺) (Calculus TWB)
    using indRelRTPO-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 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-correspondence-simulation*:
  fixes *TRel* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *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* :: ($'procT \times 'procT$) *set*
   and *Rel* :: (($'procS$, $'procT$) *Proc* $\times$ ($'procS$, $'procT$) *Proc*) *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 $\forall P\ Q\ Q'.\ (P, Q) \in TRel^+ \wedge Q \longmapsto Target* \ Q' \longrightarrow (\exists P'.\ P \longmapsto Target* \ P' \wedge (P', Q') \in TRel^+)$
  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 $TQ\ TQ'$
    assume (*TP*, *TQ*) $\in$ *TRel*
    with *target* **have** (*TargetTerm TP*, *TargetTerm TQ*) $\in$ *Rel*
     by *simp*
    moreover assume $TQ \longmapsto Target* \ TQ'$
    hence *TargetTerm TQ* $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm* TQ')
     by (*simp add*: *STCal-steps*)
    ultimately obtain P' **where** *A2*: *TargetTerm TP* $\longmapsto$(*STCal Source Target*)$*$ P'
          and *A3*: (P', *TargetTerm* TQ') $\in$ *Rel*

using *bisim*
by *blast*
from *A2* **obtain** TP' **where** $A4$: $TP \longmapsto Target* \; TP'$ **and** $A5$: $TP' \in T \; P'$
by (*auto simp add*: *STCal-steps*)
from *A3 A5 trel* **have** $(TP', TQ') \in TRel^+$
by *simp*
with *A4* **show** $\exists \, TP'. \; TP \longmapsto Target* \; TP' \wedge (TP', TQ') \in TRel^+$
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-steps*)
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-steps*)
assume $\bigwedge TQ'. \; TQ \longmapsto Target* \; TQ' \Longrightarrow \exists \, TP'. \; TP \longmapsto Target* \; TP' \wedge (TP', TQ') \in TRel^+$
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*) *indRelRTPO-impl-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \; set$
assumes *bisim*: *weak-reduction-bisimulation* (*indRelRTPO TRel*) (*STCal Source Target*)
shows *weak-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-weak-reduction-bisimulation[**where**
Rel=indRelRTPO TRel **and** *TRel=TRel*]
by *blast*

lemma (**in** *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \; set$
assumes *bisim*: *weak-reduction-bisimulation* (*indRelLTPO TRel*) (*STCal Source Target*)
shows *weak-reduction-bisimulation* $(TRel^+)$ *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 $TRel :: ('procT \times 'procT) \; set$
and $Rel :: (('procS, 'procT) \; Proc \times ('procS, 'procT) \; Proc) \; set$
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-reflC-transC-TRel-is-weak-reduction-simulation*[**where** *TRel=TRel* **and** *Rel=Rel*]
by *blast*
next
show $\forall P\ Q\ Q'.\ (P,\ Q) \in TRel^* \wedge Q \longmapsto Target{*}\ Q' \longrightarrow (\exists P'.\ P \longmapsto Target{*}\ P' \wedge (P',\ Q') \in TRel^*)$
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 $TP \longmapsto Target{*}\ TQ'$
moreover have $(TQ',\ TQ') \in TRel^*$
by *simp*
ultimately show $\exists\, TP'.\ TP \longmapsto Target{*}\ TP' \wedge (TP',\ TQ') \in TRel^*$
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-steps*)
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-steps*)
assume $\bigwedge TQ'.\ TQ \longmapsto Target{*}\ TQ' \Longrightarrow \exists\, TP'.\ TP \longmapsto Target{*}\ TP' \wedge (TP',\ TQ') \in TRel^*$
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-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 $\forall P\ Q\ Q'.\ (P,\ Q) \in TRel^+ \wedge Q \longmapsto Target\ Q' \longrightarrow (\exists P'.\ P \longmapsto Target\ P' \wedge (P',\ Q') \in TRel^+)$
  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 $TQ\ TQ'$
    assume $(TP,\ TQ) \in TRel$
    with *target* **have** $(TargetTerm\ TP,\ TargetTerm\ TQ) \in Rel$
     by *simp*
    moreover assume $TQ \longmapsto Target\ TQ'$
    hence $TargetTerm\ TQ \longmapsto(STCal\ Source\ Target)\ (TargetTerm\ TQ')$
     by (*simp add*: *STCal-step*)
    ultimately obtain P' **where** $A2$: $TargetTerm\ TP \longmapsto(STCal\ Source\ Target)\ P'$
           and $A3$: $(P',\ TargetTerm\ TQ') \in Rel$
      using *bisim*
     by *blast*
    from $A2$ **obtain** TP' **where** $A4$: $TP \longmapsto Target\ TP'$ **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** $\exists TP'.\ TP \longmapsto Target\ TP' \wedge (TP',\ TQ') \in TRel^+$
     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 $\bigwedge TQ'.\ TQ \longmapsto Target\ TQ' \Longrightarrow \exists TP'.\ TP \longmapsto Target\ TP' \wedge (TP',\ TQ') \in TRel^+$
    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*) *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* ($TRel^+$) *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* ($TRel^*$) *Target*
proof
from *bisim target trel* **show** *strong-reduction-simulation* ($TRel^*$) *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'* $\Longrightarrow$ $\exists$ *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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* × *'procT*) *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 × '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 × '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 × 'procS) set ⇒ ('procT × 'procT) set
        ⇒ ((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
    for SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  where
  encR:   (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelRST SRel TRel |
  source: (S1, S2) ∈ SRel ⟹ (SourceTerm S1, SourceTerm S2) ∈ indRelRST SRel TRel |
  target: (T1, T2) ∈ TRel ⟹ (TargetTerm T1, TargetTerm T2) ∈ indRelRST SRel TRel

abbreviation (in encoding) indRelRSTinfix
    :: ('procS, 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
        ⇒ ('procS, 'procT) Proc ⇒ bool (‹- 𝓡⟦·⟧R<-,-> -› [75, 75, 75, 75] 80)
```

where
P ℛ⟦·⟧R<SRel,TRel> *Q* ≡ (*P*, *Q*) ∈ *indRelRST SRel TRel*

inductive-set (**in** *encoding*) *indRelRSTPO*
:: ('*procS* × '*procS*) *set* ⇒ ('*procT* × '*procT*) *set*
⇒ ((('*procS*, '*procT*) *Proc*) × (('*procS*, '*procT*) *Proc*)) *set*
for *SRel* :: ('*procS* × '*procS*) *set*
and *TRel* :: ('*procT* × '*procT*) *set*
where
encR: (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelRSTPO SRel TRel* |
source: (*S1*, *S2*) ∈ *SRel* ⟹ (*SourceTerm S1*, *SourceTerm S2*) ∈ *indRelRSTPO SRel TRel* |
target: (*T1*, *T2*) ∈ *TRel* ⟹ (*TargetTerm T1*, *TargetTerm T2*) ∈ *indRelRSTPO SRel TRel* |
trans: ⟦(*P*, *Q*) ∈ *indRelRSTPO SRel TRel*; (*Q*, *R*) ∈ *indRelRSTPO SRel TRel*⟧
⟹ (*P*, *R*) ∈ *indRelRSTPO SRel TRel*

abbreviation (**in** *encoding*) *indRelRSTPOinfix* ::
('*procS*, '*procT*) *Proc* ⇒ ('*procS* × '*procS*) *set* ⇒ ('*procT* × '*procT*) *set*
⇒ ('*procS*, '*procT*) *Proc* ⇒ *bool* (‹- ≲⟦·⟧R<-,-> -› [75, 75, 75, 75] 80)
where
P ≲⟦·⟧R<SRel,TRel> *Q* ≡ (*P*, *Q*) ∈ *indRelRSTPO SRel TRel*

lemma (**in** *encoding*) *indRelRSTPO-refl*:
fixes *SRel* :: ('*procS* × '*procS*) *set*
and *TRel* :: ('*procT* × '*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* ≲⟦·⟧R<SRel,TRel> *P*
proof (*cases P*)
case (*SourceTerm SP*)
assume *SP* ∈*S P*
with *reflS* **show** *P* ≲⟦·⟧R<SRel,TRel> *P*
unfolding *refl-on-def*
by (*simp add*: *indRelRSTPO.source*)
next
case (*TargetTerm TP*)
assume *TP* ∈*T P*
with *reflT* **show** *P* ≲⟦·⟧R<SRel,TRel> *P*
unfolding *refl-on-def*
by (*simp add*: *indRelRSTPO.target*)
qed
qed

lemma (**in** *encoding*) *indRelRSTPO-trans*:
fixes *SRel* :: ('*procS* × '*procS*) *set*
and *TRel* :: ('*procT* × '*procT*) *set*
shows *trans* (*indRelRSTPO SRel TRel*)
unfolding *trans-def*
proof *clarify*
fix *P Q R*
assume *P* ≲⟦·⟧R<SRel,TRel> *Q* **and** *Q* ≲⟦·⟧R<SRel,TRel> *R*
thus *P* ≲⟦·⟧R<SRel,TRel> *R*
by (*rule indRelRSTPO.trans*)
qed

lemma (**in** *encoding*) *refl-trans-closure-of-indRelRST*:
fixes *SRel* :: ('*procS* × '*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⟦·⟧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 ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
        ⇒ ('procS, 'procT) Proc ⇒ bool (‹- ℛ⟦·⟧L<-,-> -› [75, 75, 75, 75] 80)
  where
  P ℛ⟦·⟧L<SRel,TRel> Q ≡ (P, Q) ∈ indRelLST SRel TRel

inductive-set (in encoding) indRelLSTPO
    :: ('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) ∈ indRelLSTPO SRel TRel |
  source: (S1, S2) ∈ SRel ⟹ (SourceTerm S1, SourceTerm S2) ∈ indRelLSTPO SRel TRel |
  target: (T1, T2) ∈ TRel ⟹ (TargetTerm T1, TargetTerm T2) ∈ indRelLSTPO SRel TRel |
  trans:  ⟦(P, Q) ∈ indRelLSTPO SRel TRel; (Q, R) ∈ indRelLSTPO SRel TRel⟧
          ⟹ (P, R) ∈ indRelLSTPO SRel TRel

abbreviation (in encoding) indRelLSTPOinfix
    :: ('procS, 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
        ⇒ ('procS, 'procT) Proc ⇒ bool (‹- ≲⟦·⟧L<-,-> -› [75, 75, 75, 75] 80)
  where
  P ≲⟦·⟧L<SRel,TRel> Q ≡ (P, Q) ∈ indRelLSTPO SRel TRel

lemma (in encoding) indRelLSTPO-refl:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × '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 ≲⟦·⟧L<SRel,TRel> P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈S P
    with reflS show P ≲⟦·⟧L<SRel,TRel> P
        unfolding refl-on-def
      by (simp add: indRelLSTPO.source)
  next
    case (TargetTerm TP)
    assume TP ∈T P
    with reflT show P ≲⟦·⟧L<SRel,TRel> P
        unfolding refl-on-def
      by (simp add: indRelLSTPO.target)
  qed
qed

lemma (in encoding) indRelLSTPO-trans:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  shows trans (indRelLSTPO SRel TRel)
    unfolding trans-def
proof clarify
  fix P Q R
  assume P ≲⟦·⟧L<SRel,TRel> Q and Q ≲⟦·⟧L<SRel,TRel> R
  thus P ≲⟦·⟧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 ≲⟦·⟧L<SRel,TRel> Q
  thus (P, Q) ∈ (indRelLST SRel TRel)*
  proof induct
    case (encL S)
    show (TargetTerm (⟦S⟧), SourceTerm S) ∈ (indRelLST SRel TRel)*
        using indRelLST.encL[of S SRel TRel]
      by simp
  next
    case (source S1 S2)
    assume (S1, S2) ∈ SRel
    thus (SourceTerm S1, SourceTerm S2) ∈ (indRelLST SRel TRel)*
        using indRelLST.source[of S1 S2 SRel TRel]
      by simp
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    thus (TargetTerm T1, TargetTerm T2) ∈ (indRelLST SRel TRel)*
        using indRelLST.target[of T1 T2 TRel SRel]
      by simp
  next
    case (trans P Q R)
    assume (P, Q) ∈ (indRelLST SRel TRel)* and (Q, R) ∈ (indRelLST SRel TRel)*
    thus (P, R) ∈ (indRelLST SRel TRel)*
      by simp
  qed
next
  fix P Q
  assume (P, Q) ∈ (indRelLST SRel TRel)*
  thus P ≲⟦·⟧L<SRel,TRel> Q
  proof induct
    from reflS reflT show  P ≲⟦·⟧L<SRel,TRel> P
        using indRelLSTPO-refl[of SRel TRel]
        unfolding refl-on-def
      by simp
  next
    case (step Q R)
    assume P ≲⟦·⟧L<SRel,TRel> Q
    moreover assume Q ℛ⟦·⟧L<SRel,TRel> R
    hence Q ≲⟦·⟧L<SRel,TRel> R
      by (induct, simp-all add: indRelLSTPO.intros)
    ultimately show P ≲⟦·⟧L<SRel,TRel> R
      by (rule indRelLSTPO.trans)
  qed
qed

inductive-set (in encoding) indRelST
    :: ('procS × 'procS) set ⇒ ('procT × 'procT) set
      ⇒ ((('procS, 'procT) Proc) × (('procS, 'procT) Proc)) set
    for SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
  where
  encR:   (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelST SRel TRel |
  encL:   (TargetTerm (⟦S⟧), SourceTerm S) ∈ indRelST SRel TRel |
  source: (S1, S2) ∈ SRel ⟹ (SourceTerm S1, SourceTerm S2) ∈ indRelST SRel TRel |
```

target: (*T1*, *T2*) ∈ *TRel* ⟹ (*TargetTerm T1*, *TargetTerm T2*) ∈ *indRelST SRel TRel*

abbreviation (**in** *encoding*) *indRelSTinfix*
:: ('*procS*, '*procT*) *Proc* ⇒ ('*procS* × '*procS*) *set* ⇒ ('*procT* × '*procT*) *set*
⇒ ('*procS*, '*procT*) *Proc* ⇒ *bool* (‹- ℛ⟦·⟧<-,-> -› [*75*, *75*, *75*, *75*] *80*)
where
P ℛ⟦·⟧<*SRel*,*TRel*> *Q* ≡ (*P*, *Q*) ∈ *indRelST SRel TRel*

lemma (**in** *encoding*) *indRelST-symm*:
fixes *SRel* :: ('*procS* × '*procS*) *set*
and *TRel* :: ('*procT* × '*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*) ∈ *indRelST SRel TRel*
thus (*Q*, *P*) ∈ *indRelST SRel TRel*
proof *induct*
case (*encR S*)
show (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *indRelST SRel TRel*
by (*rule indRelST*.*encL*)
next
case (*encL S*)
show (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelST SRel TRel*
by (*rule indRelST*.*encR*)
next
case (*source S1 S2*)
assume (*S1*, *S2*) ∈ *SRel*
with *symmS* **show** (*SourceTerm S2*, *SourceTerm S1*) ∈ *indRelST SRel TRel*
unfolding *sym-def*
by (*simp add*: *indRelST*.*source*)
next
case (*target 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

inductive-set (**in** *encoding*) *indRelSTEQ*
:: ('*procS* × '*procS*) *set* ⇒ ('*procT* × '*procT*) *set*
⇒ ((('*procS*, '*procT*) *Proc*) × (('*procS*, '*procT*) *Proc*)) *set*
for *SRel* :: ('*procS* × '*procS*) *set*
and *TRel* :: ('*procT* × '*procT*) *set*
where
encR: (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelSTEQ SRel TRel* |
encL: (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *indRelSTEQ SRel TRel* |
source: (*S1*, *S2*) ∈ *SRel* ⟹ (*SourceTerm S1*, *SourceTerm S2*) ∈ *indRelSTEQ SRel TRel* |
target: (*T1*, *T2*) ∈ *TRel* ⟹ (*TargetTerm T1*, *TargetTerm T2*) ∈ *indRelSTEQ SRel TRel* |
trans: ⟦(*P*, *Q*) ∈ *indRelSTEQ SRel TRel*; (*Q*, *R*) ∈ *indRelSTEQ SRel TRel*⟧
⟹ (*P*, *R*) ∈ *indRelSTEQ SRel TRel*

abbreviation (**in** *encoding*) *indRelSTEQinfix*
:: ('*procS*, '*procT*) *Proc* ⇒ ('*procS* × '*procS*) *set* ⇒ ('*procT* × '*procT*) *set*
⇒ ('*procS*, '*procT*) *Proc* ⇒ *bool* (‹- ∼⟦·⟧<-,-> -› [*75*, *75*, *75*, *75*] *80*)
where
P ∼⟦·⟧<*SRel*,*TRel*> *Q* ≡ (*P*, *Q*) ∈ *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 ∼⟦·⟧<SRel,TRel> P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈S P
    moreover have SourceTerm SP ∼⟦·⟧<SRel,TRel> TargetTerm (⟦SP⟧)
      by (rule indRelSTEQ.encR)
    moreover have TargetTerm (⟦SP⟧) ∼⟦·⟧<SRel,TRel> SourceTerm SP
      by (rule indRelSTEQ.encL)
    ultimately show P ∼⟦·⟧<SRel,TRel> P
      by (simp add: indRelSTEQ.trans[where P=SourceTerm SP and Q=TargetTerm (⟦SP⟧)])
  next
    case (TargetTerm TP)
    assume TP ∈T P
    with reflT show P ∼⟦·⟧<SRel,TRel> 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 ∼⟦·⟧<SRel,TRel> Q
  thus Q ∼⟦·⟧<SRel,TRel> P
  proof induct
    case (encR S)
    show TargetTerm (⟦S⟧) ∼⟦·⟧<SRel,TRel> SourceTerm S
      by (rule indRelSTEQ.encL)
  next
    case (encL S)
    show SourceTerm S ∼⟦·⟧<SRel,TRel> TargetTerm (⟦S⟧)
      by (rule indRelSTEQ.encR)
  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 $R \sim[\![\cdot]\!]{<}SRel,TRel{>}\ P$
    by (*rule indRelSTEQ.trans*)
  qed
qed

lemma (**in** *encoding*) *indRelSTEQ-trans*:
  fixes *SRel* :: ('*procS* × '*procS*) *set*
    and *TRel* :: ('*procT* × '*procT*) *set*
  shows *trans* (*indRelSTEQ SRel TRel*)
    unfolding *trans-def*
proof *clarify*
  fix *P Q R*
  assume $P \sim[\![\cdot]\!]{<}SRel,TRel{>}\ Q$ **and** $Q \sim[\![\cdot]\!]{<}SRel,TRel{>}\ R$
  thus $P \sim[\![\cdot]\!]{<}SRel,TRel{>}\ R$
    by (*rule indRelSTEQ.trans*)
qed

lemma (**in** *encoding*) *refl-trans-closure-of-indRelST*:
  fixes *SRel* :: ('*procS* × '*procS*) *set*
    and *TRel* :: ('*procT* × '*procT*) *set*
  assumes *reflT*: *refl TRel*
  shows *indRelSTEQ SRel TRel* = (*indRelST SRel TRel*)*
proof *auto*
  fix *P Q*
  assume $P \sim[\![\cdot]\!]{<}SRel,TRel{>}\ Q$
  thus (*P*, *Q*) ∈ (*indRelST SRel TRel*)*
  proof *induct*
    case (*encR S*)
    show (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ (*indRelST SRel TRel*)*
        using *indRelST.encR*[*of S SRel TRel*]
      by *simp*
  next
    case (*encL S*)
    show (*TargetTerm* ($[\![S]\!]$), *SourceTerm S*) ∈ (*indRelST SRel TRel*)*
        using *indRelST.encL*[*of S SRel TRel*]
      by *simp*
  next
    case (*source S1 S2*)
    assume (*S1*, *S2*) ∈ *SRel*
    thus (*SourceTerm S1*, *SourceTerm S2*) ∈ (*indRelST SRel TRel*)*
        using *indRelST.source*[*of S1 S2 SRel TRel*]
      by *simp*
  next
    case (*target T1 T2*)
    assume (*T1*, *T2*) ∈ *TRel*
    thus (*TargetTerm T1*, *TargetTerm T2*) ∈ (*indRelST SRel TRel*)*
        using *indRelST.target*[*of T1 T2 TRel SRel*]
      by *simp*
  next
    case (*trans P Q R*)
    assume (*P*, *Q*) ∈ (*indRelST SRel TRel*)* **and** (*Q*, *R*) ∈ (*indRelST SRel TRel*)*
    thus (*P*, *R*) ∈ (*indRelST SRel TRel*)*
      by *simp*
  qed
next
  fix *P Q*
  assume (*P*, *Q*) ∈ (*indRelST SRel TRel*)*
  thus $P \sim[\![\cdot]\!]{<}SRel,TRel{>}\ Q$
  proof *induct*
    from *reflT* **show** $P \sim[\![\cdot]\!]{<}SRel,TRel{>}\ P$
        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 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 indRelST SRel TRel = symcl (indRelRST SRel TRel)
    and indRelSTEQ SRel TRel = (symcl ((indRelRST SRel TRel)⁼))⁺
proof −
  show indRelST SRel TRel = symcl (indRelRST SRel TRel)
  proof auto
    fix P Q
    assume P 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 ℛ⟦·⟧<SRel,TRel> Q
    proof (auto simp add: symcl-def indRelRST.simps)
      fix S
      show SourceTerm S ℛ⟦·⟧<SRel,TRel> TargetTerm (⟦S⟧)
        by (rule indRelST.encR)
    next
      fix S1 S2
      assume (S1, S2) ∈ SRel
      thus SourceTerm S1 ℛ⟦·⟧<SRel,TRel> SourceTerm S2
        by (rule indRelST.source)
    next
```

```
      fix T1 T2
      assume (T1, T2) ∈ TRel
      thus TargetTerm T1 R⟦·⟧<SRel,TRel> TargetTerm T2
        by (rule indRelST.target)
    next
      fix S
      show TargetTerm (⟦S⟧) R⟦·⟧<SRel,TRel> SourceTerm S
        by (rule indRelST.encL)
    next
      fix S1 S2
      assume (S1, S2) ∈ SRel
      with symmS show SourceTerm S2 R⟦·⟧<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 R⟦·⟧<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 R⟦·⟧<SRel,TRel> Q
    proof (auto simp add: symcl-def indRelLST.simps)
      fix S
      show SourceTerm S R⟦·⟧<SRel,TRel> TargetTerm (⟦S⟧)
        by (rule indRelST.encR)
    next
      fix S1 S2
      assume (S1, S2) ∈ SRel
      thus SourceTerm S1 R⟦·⟧<SRel,TRel> SourceTerm S2
        by (rule indRelST.source)
    next
      fix T1 T2
      assume (T1, T2) ∈ TRel
      thus TargetTerm T1 R⟦·⟧<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
    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
    case (source S1 S2)
```

```
    assume (S1, S2) ∈ SRel
    thus (SourceTerm S1, SourceTerm S2) ∈ (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) ∈ TRel
    thus (TargetTerm T1, TargetTerm T2) ∈ (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) ∈ (symcl (indRelLSTPO SRel TRel))^+
       and (Q, R) ∈ (symcl (indRelLSTPO SRel TRel))^+
    thus (P, R) ∈ (symcl (indRelLSTPO SRel TRel))^+
      by simp
  qed
next
  fix P Q
  assume (P, Q) ∈ (symcl (indRelLSTPO SRel TRel))^+
  thus P ∼⟦·⟧<SRel,TRel> Q
  proof induct
    fix Q
    assume (P, Q) ∈ symcl (indRelLSTPO SRel TRel)
    thus P ∼⟦·⟧<SRel,TRel> Q
      unfolding symcl-def
    proof auto
      assume P ≲⟦·⟧L<SRel,TRel> Q
      thus P ∼⟦·⟧<SRel,TRel> Q
      proof (induct, simp add: indRelSTEQ.encL, 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
    next
      assume Q ≲⟦·⟧L<SRel,TRel> P
      hence Q ∼⟦·⟧<SRel,TRel> P
      proof (induct, simp add: indRelSTEQ.encL, 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 P ∼⟦·⟧<SRel,TRel> Q
          using indRelSTEQ-symm[of SRel TRel]
          unfolding sym-def
        by blast
    qed
  next
    case (step Q R)
    assume P ∼⟦·⟧<SRel,TRel> Q
    moreover assume (Q, R) ∈ symcl (indRelLSTPO SRel TRel)
    hence Q ∼⟦·⟧<SRel,TRel> R
      unfolding symcl-def
    proof auto
      assume Q ≲⟦·⟧L<SRel,TRel> R
```

```
      thus Q ∼⟦·⟧<SRel,TRel> R
      proof (induct, simp add: indRelSTEQ.encL, 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
    next
      assume R ≲⟦·⟧L<SRel,TRel> Q
      hence R ∼⟦·⟧<SRel,TRel> Q
      proof (induct, simp add: indRelSTEQ.encL, 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
```

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 R⟦·⟧R<SRel,TRel> 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 R⟦·⟧R<SRel,TRel> 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 R⟦·⟧L<SRel,TRel> 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 $TRel$ :: $('procT \times 'procT)$ 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** $encoding$) $indRelST\text{-}to\text{-}SRel$:
fixes $SRel$ :: $('procS \times 'procS)$ set
and $TRel$ :: $('procT \times 'procT)$ 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** $encoding$) $indRelST\text{-}to\text{-}TRel$:
fixes $SRel$ :: $('procS \times 'procS)$ set
and $TRel$ :: $('procT \times 'procT)$ 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** $encoding$) $indRelRSTPO\text{-}to\text{-}SRel\text{-}and\text{-}TRel$:
fixes $SRel$ :: $('procS \times 'procS)$ set
and $TRel$ :: $('procT \times 'procT)$ 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 $\forall TP\ SQ.\ TP \in T\ SourceTerm\ S \wedge SQ \in S\ TargetTerm\ ([\![S]\!]) \longrightarrow False$
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 SourceTerm S1* $\wedge$ *SQ* $\in$*S SourceTerm S2* $\longrightarrow$ *False*
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*
$\longrightarrow$ ($\exists$ *S*. (*SP*, *S*) $\in$ *SRel** $\wedge$ ($[\![S]\!]$, *TQ*) $\in$ *TRel**)
and $\forall$ *TP SQ. TP* $\in$*T TargetTerm T1* $\wedge$ *SQ* $\in$*S TargetTerm T2* $\longrightarrow$ *False*
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* $\longrightarrow$ ($\exists$ *S*. (*SP*, *S*) $\in$ *SRel** $\wedge$ ($[\![S]\!]$, *TQ*) $\in$ *TRel**)
and *A3*: $\forall$ *TP SQ. TP* $\in$*T P* $\wedge$ *SQ* $\in$*S Q* $\longrightarrow$ *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* $\longrightarrow$ *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** $\wedge$ ($[\![S]\!]$, *TR*) $\in$ *TRel**

```
      by blast
    next
      case (TargetTerm TQ)
      assume A11: TQ ∈T Q
      with A2 A9 obtain S where A12: (SP, S) ∈ SRel*
                           and A13: (⟦S⟧, TQ) ∈ TRel*
        by blast
      from A8 A10 A11 have (TQ, TR) ∈ TRel*
        by simp
      with A13 have (⟦S⟧, TR) ∈ TRel*
        by simp
      with A12 show ∃ S. (SP, S) ∈ SRel* ∧ (⟦S⟧, TR) ∈ TRel*
        by blast
    qed
  qed
  show ∀ TP SR. TP ∈T P ∧ SR ∈S R ⟶ False
  proof clarify
    fix TP SR
    assume A9: TP ∈T P and A10: SR ∈S R
    show False
    proof (cases Q)
      case (SourceTerm SQ)
      assume SQ ∈S Q
      with A3 A9 show False
        by blast
    next
      case (TargetTerm TQ)
      assume TQ ∈T Q
      with A7 A10 show False
        by blast
    qed
  qed
  show ∀ TP TR. TP ∈T P ∧ TR ∈T R ⟶ (TP, TR) ∈ TRel+
  proof clarify
    fix TP TR
    assume A9: TP ∈T P and A10: TR ∈T R
    show (TP, TR) ∈ TRel+
    proof (cases Q)
      case (SourceTerm SQ)
      assume SQ ∈S Q
      with A3 A9 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

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 ≲⟦·⟧L<SRel,TRel> Q
  shows ∀ SP SQ. SP ∈S P ∧ SQ ∈S Q ⟶ (SP, SQ) ∈ SRel+
```

and $\forall SP\ TQ.\ SP \in S\ P \wedge TQ \in T\ Q \longrightarrow False$
and $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q \longrightarrow (\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SQ) \in SRel^*)$
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 $\forall SP\ SQ.\ SP \in S\ TargetTerm\ ([\![S]\!]) \wedge SQ \in S\ SourceTerm\ S \longrightarrow (SP, SQ) \in SRel^+$
and $\forall SP\ TQ.\ SP \in S\ TargetTerm\ ([\![S]\!]) \wedge TQ \in T\ SourceTerm\ S \longrightarrow False$
and $\forall TP\ TQ.\ TP \in T\ TargetTerm\ ([\![S]\!]) \wedge TQ \in T\ SourceTerm\ S \longrightarrow (TP, TQ) \in TRel^+$
by *simp+*
have $([\![S]\!], [\![S]\!]) \in TRel^*$
by *simp*
moreover have $(S, S) \in SRel^*$
by *simp*
ultimately show $\forall TP\ SQ.\ TP \in T\ TargetTerm\ ([\![S]\!]) \wedge SQ \in S\ SourceTerm\ S \longrightarrow$
$(\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SQ) \in SRel^*)$
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 $\forall TP\ SQ.\ TP \in T\ SourceTerm\ S1 \wedge SQ \in S\ SourceTerm\ S2$
$\longrightarrow (\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SQ) \in SRel^*)$
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 $\forall TP\ SQ.\ TP \in T\ TargetTerm\ T1 \wedge SQ \in S\ TargetTerm\ T2$
$\longrightarrow (\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SQ) \in SRel^*)$
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 $A3$: $\forall TP\ SQ.\ TP \in T\ P \wedge SQ \in S\ Q$
$\longrightarrow (\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SQ) \in SRel^*)$
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 $A7$: $\forall TQ\ SR.\ TQ \in T\ Q \wedge SR \in S\ R \longrightarrow (\exists S.\ (TQ, [\![S]\!]) \in TRel^* \wedge (S, SR) \in SRel^*)$
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 $\forall TP\ SR.\ TP \in T\ P \wedge SR \in S\ R \longrightarrow (\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SR) \in SRel^*)$
proof *clarify*
fix *TP SR*
assume *A9*: $TP \in T\ P$ **and** *A10*: $SR \in S\ R$
show $\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SR) \in SRel^*$
proof (*cases Q*)
case (*SourceTerm SQ*)
assume *A11*: $SQ \in S\ Q$
with *A3 A9* **obtain** *S* **where** *A12*: $(TP, [\![S]\!]) \in TRel^*$ **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** $\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SR) \in SRel^*$
by *blast*
next
case (*TargetTerm TQ*)
assume *A11*: $TQ \in T\ Q$
with *A7 A10* **obtain** *S* **where** *A12*: $(TQ, [\![S]\!]) \in TRel^*$ **and** *A13*: $(S, SR) \in SRel^*$
by *blast*
from *A4 A9 A11* **have** $(TP, TQ) \in TRel^*$
by *simp*
from *this A12* **have** $(TP, [\![S]\!]) \in TRel^*$
by *simp*
with *A13* **show** $\exists S.\ (TP, [\![S]\!]) \in TRel^* \wedge (S, SR) \in SRel^*$
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 indRelRSTPO, indRelLSTPO, or 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↓<SWB>a*
hence *SourceTerm SQ↓.a*
by *simp*
ultimately have *SourceTerm SP↓.a*
using *reflection reflection-of-barbs-in-barbed-encoding*[**where** *Rel=Rel*]
by *blast*
thus *SP↓<SWB>a*
by *simp*
qed

lemma (**in** *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-reflect-barbs*:
fixes *SRel* :: (*'procS* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *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* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *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* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *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 **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* × ′*procS*) *set*
and *Rel* :: ((′*procS*, ′*procT*) *Proc* × (′*procS*, ′*procT*) *Proc*) *set*
assumes *reflection*: *rel-weakly-reflects-barbs Rel* (*STCalWB SWB TWB*)
and *sourceInRel*: ∀ *S1 S2*. (*S1*, *S2*) ∈ *SRel* ⟶ (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
shows *rel-weakly-reflects-barbs SRel SWB*
proof *clarify*
fix *SP SQ a SQ′*
assume (*SP*, *SQ*) ∈ *SRel*
with *sourceInRel* **have** (*SourceTerm SP*, *SourceTerm SQ*) ∈ *Rel*
by *blast*
moreover assume *SQ* ⟼(*Calculus SWB*)∗ *SQ′* **and** *SQ′*↓<*SWB*>*a*
hence *SourceTerm SQ*⇓.*a*
by *blast*
ultimately have *SourceTerm SP*⇓.*a*
using *reflection weak-reflection-of-barbs-in-barbed-encoding*[**where** *Rel*=*Rel*]
by *blast*
thus *SP*⇓<*SWB*>*a*
by *simp*
qed

lemma (**in** *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs*:
fixes *SRel* :: (′*procS* × ′*procS*) *set*
and *TRel* :: (′*procT* × ′*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* **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* **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* × ′*procS*) *set*
and *TRel* :: (′*procT* × ′*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* **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* **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 × 'procS) set
    and TRel :: ('procT × 'procT) 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 ind relRTPO is a subrelation of indRelTEQ. If SRel is reflexive then indRelRTPO is a subrelation of indRelRTPO. Moreover, indRelRSTPO is a subrelation of indRelSTEQ.

```
lemma (in encoding) indRelRTPO-to-indRelTEQ:
  fixes TRel :: ('procT × 'procT) set
    and P Q  :: ('procS, 'procT) Proc
  assumes rel:   P ≲⟦·⟧RT<TRel> Q
      and reflT: refl TRel
  shows P ∼⟦·⟧T<TRel> Q
    using rel
proof induct
  case (encR S)
  show SourceTerm S ∼⟦·⟧T<TRel> TargetTerm (⟦S⟧)
    by (rule indRelTEQ.encR)
next
  case (source S)
  from reflT show SourceTerm S ∼⟦·⟧T<TRel> SourceTerm S
      using indRelTEQ-refl[of TRel]
      unfolding refl-on-def
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1 ∼⟦·⟧T<TRel> TargetTerm T2
    by (rule indRelTEQ.target)
next
  case (trans TP TQ TR)
  assume TP ∼⟦·⟧T<TRel> TQ and TQ ∼⟦·⟧T<TRel> TR
  thus TP ∼⟦·⟧T<TRel> TR
    by (rule indRelTEQ.trans)
qed

lemma (in encoding) indRelRTPO-to-indRelRSTPO:
  fixes SRel :: ('procS × 'procS) set
    and TRel :: ('procT × 'procT) set
    and P Q  :: ('procS, 'procT) Proc
  assumes rel:   P ≲⟦·⟧RT<TRel> Q
```

and *reflS*: *refl SRel*
shows $P \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q$
using *rel*
proof *induct*
case (*encR S*)
show $SourceTerm\ S \lesssim[\![\cdot]\!]R<SRel,TRel>\ TargetTerm\ ([\![S]\!])$
by (*rule indRelRSTPO.encR*)
next
case (*source S*)
from *reflS* **show** $SourceTerm\ S \lesssim[\![\cdot]\!]R<SRel,TRel>\ SourceTerm\ S$
unfolding *refl-on-def*
by (*simp add*: *indRelRSTPO.source*)
next
case (*target T1 T2*)
assume $(T1,\ T2) \in TRel$
thus $TargetTerm\ T1 \lesssim[\![\cdot]\!]R<SRel,TRel>\ TargetTerm\ T2$
by (*rule indRelRSTPO.target*)
next
case (*trans 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*) *indRelRSTPO-to-indRelSTEQ*:
fixes *SRel* :: (*'procS* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *set*
and *P Q* :: (*'procS*, *'procT*) *Proc*
assumes *rel*: $P \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q$
shows $P \sim[\![\cdot]\!]<SRel,TRel>\ Q$
using *rel*
proof *induct*
case (*encR S*)
show $SourceTerm\ S \sim[\![\cdot]\!]<SRel,TRel>\ TargetTerm\ ([\![S]\!])$
by (*rule indRelSTEQ.encR*)
next
case (*source S1 S2*)
assume $(S1,\ S2) \in SRel$
thus $SourceTerm\ S1 \sim[\![\cdot]\!]<SRel,TRel>\ SourceTerm\ S2$
by (*rule indRelSTEQ.source*)
next
case (*target T1 T2*)
assume $(T1,\ T2) \in TRel$
thus $TargetTerm\ T1 \sim[\![\cdot]\!]<SRel,TRel>\ TargetTerm\ T2$
by (*rule indRelSTEQ.target*)
next
case (*trans P Q R*)
assume $P \sim[\![\cdot]\!]<SRel,TRel>\ Q$ **and** $Q \sim[\![\cdot]\!]<SRel,TRel>\ R$
thus $P \sim[\![\cdot]\!]<SRel,TRel>\ R$
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 $P \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q$ **and** $P \longmapsto(STCal\ Source\ Target)*\ P'$
thus $\exists\, Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
proof (*induct arbitrary*: P')
case (*encR S*)
have $SourceTerm\ S \lesssim[\![\cdot]\!]RT<TRel>\ TargetTerm\ ([\![S]\!])$
by (*rule indRelRTPO.encR*)
moreover assume $SourceTerm\ S \longmapsto(STCal\ Source\ Target)*\ P'$
ultimately obtain Q' **where** $A1$: $TargetTerm\ ([\![S]\!]) \longmapsto(STCal\ Source\ Target)*\ Q'$
and $A2$: $P' \lesssim[\![\cdot]\!]RT<TRel>\ Q'$
using *bisimT*
by *blast*
from *reflS A2* **have** $P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by (*simp add*: *indRelRTPO-to-indRelRSTPO*)
with *A1* **show** $\exists\, Q'.\ TargetTerm\ ([\![S]\!]) \longmapsto(STCal\ Source\ Target)*\ Q' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by *blast*
next
case (*source S1 S2*)
assume $SourceTerm\ S1 \longmapsto(STCal\ Source\ Target)*\ P'$
from *this* **obtain** $S1'$ **where** $B1$: $S1' \in S\ P'$ **and** $B2$: $S1 \longmapsto Source*\ S1'$
by (*auto simp add*: *STCal-steps*(*1*))
assume $(S1,\ S2) \in SRel$
with *B2 bisimS* **obtain** $S2'$ **where** $B3$: $S2 \longmapsto Source*\ S2'$ **and** $B4$: $(S1',\ S2') \in SRel$
by *blast*
from *B3* **have** $SourceTerm\ S2 \longmapsto(STCal\ Source\ Target)*\ (SourceTerm\ S2')$
by (*simp add*: *STCal-steps*(*1*))
moreover from *B1 B4* **have** $P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ SourceTerm\ S2'$
by (*simp add*: *indRelRSTPO.source*)
ultimately show $\exists\, Q'.\ SourceTerm\ S2 \longmapsto(STCal\ Source\ Target)*\ Q' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by *blast*
next
case (*target T1 T2*)
assume $(T1,\ T2) \in TRel$
hence $TargetTerm\ T1 \lesssim[\![\cdot]\!]RT<TRel>\ TargetTerm\ T2$
by (*rule indRelRTPO.target*)
moreover assume $TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)*\ P'$
ultimately obtain Q' **where** $C1$: $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)*\ Q'$
and $C2$: $P' \lesssim[\![\cdot]\!]RT<TRel>\ Q'$
using *bisimT*
by *blast*
from *reflS C2* **have** $P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by (*simp add*: *indRelRTPO-to-indRelRSTPO*)
with *C1* **show** $\exists\, Q'.\ TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)*\ Q' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by *blast*
next
case (*trans P Q R*)
assume $P \longmapsto(STCal\ Source\ Target)*\ P'$
and $\bigwedge P'.\ P \longmapsto(STCal\ Source\ Target)*\ P'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
from *this* **obtain** Q' **where** $D1$: $Q \longmapsto(STCal\ Source\ Target)*\ Q'$ **and** $D2$: $P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ Q'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q'$
$\Longrightarrow \exists\, R'.\ R \longmapsto(STCal\ Source\ Target)*\ R' \wedge Q' \lesssim[\![\cdot]\!]R<SRel,TRel>\ R'$
with *D1* **obtain** R' **where** $D3$: $R \longmapsto(STCal\ Source\ Target)*\ R'$ **and** $D4$: $Q' \lesssim[\![\cdot]\!]R<SRel,TRel>\ R'$
by *blast*
from *D2 D4* **have** $P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ R'$
by (*rule indRelRSTPO.trans*)
with *D3* **show** $\exists\, R'.\ R \longmapsto(STCal\ Source\ Target)*\ R' \wedge P' \lesssim[\![\cdot]\!]R<SRel,TRel>\ R'$
by *blast*

qed
next
fix $P\ Q\ Q'$
assume $P \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q$ **and** $Q \longmapsto(STCal\ Source\ Target)*\ Q'$
thus $\exists P'.\ P \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
proof (*induct arbitrary*: Q')
case (*encR S*)
have $SourceTerm\ S \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ ([\![S]\!])$
by (*rule indRelRTPO.encR*)
moreover assume $TargetTerm\ ([\![S]\!]) \longmapsto(STCal\ Source\ Target)*\ Q'$
ultimately obtain P' **where** *E1*: $SourceTerm\ S \longmapsto(STCal\ Source\ Target)*\ P'$
and *E2*: $P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
using *bisimT*
by *blast*
from *reflS E2* **have** $P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by (*simp add*: *indRelRTPO-to-indRelRSTPO*)
with *E1* **show** $\exists P'.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by *blast*
next
case (*source S1 S2*)
assume $SourceTerm\ S2 \longmapsto(STCal\ Source\ Target)*\ Q'$
from *this* **obtain** $S2'$ **where** *F1*: $S2' \in S\ Q'$ **and** *F2*: $S2 \longmapsto Source*\ S2'$
by (*auto simp add*: *STCal-steps*(*1*))
assume $(S1,\ S2) \in SRel$
with *F2 bisimS* **obtain** $S1'$ **where** *F3*: $S1 \longmapsto Source*\ S1'$ **and** *F4*: $(S1',\ S2') \in SRel$
by *blast*
from *F3* **have** $SourceTerm\ S1 \longmapsto(STCal\ Source\ Target)*\ (SourceTerm\ S1')$
by (*simp add*: *STCal-steps*(*1*))
moreover from *F1 F4* **have** $SourceTerm\ S1' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by (*simp add*: *indRelRSTPO.source*)
ultimately show $\exists P'.\ SourceTerm\ S1 \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by *blast*
next
case (*target T1 T2*)
assume $(T1,\ T2) \in TRel$
hence $TargetTerm\ T1 \lesssim[\![\cdot]\!]RT{<}TRel{>}\ TargetTerm\ T2$
by (*rule indRelRTPO.target*)
moreover assume $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)*\ Q'$
ultimately obtain P' **where** *G1*: $TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)*\ P'$
and *G2*: $P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
using *bisimT*
by *blast*
from *reflS G2* **have** $P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by (*simp add*: *indRelRTPO-to-indRelRSTPO*)
with *G1* **show** $\exists P'.\ TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by *blast*
next
case (*trans P Q R R'*)
assume $R \longmapsto(STCal\ Source\ Target)*\ R'$
and $\bigwedge R'.\ R \longmapsto(STCal\ Source\ Target)*\ R'$
$\Longrightarrow \exists Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q' \land Q' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R'$
from *this* **obtain** Q' **where** *H1*: $Q \longmapsto(STCal\ Source\ Target)*\ Q'$ **and** *H2*: $Q' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q'$
$\Longrightarrow \exists P'.\ P \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
with *H1* **obtain** P' **where** *H3*: $P \longmapsto(STCal\ Source\ Target)*\ P'$ **and** *H4*: $P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q'$
by *blast*
from *H4 H2* **have** $P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R'$
by (*rule indRelRSTPO.trans*)
with *H3* **show** $\exists P'.\ P \longmapsto(STCal\ Source\ Target)*\ P' \land P' \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R'$
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 respection 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* $(\lambda P\ a.\ a \in Barbs \wedge P{\Downarrow}.a)$

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* $(\lambda P\ a.\ a \in Barbs \wedge P{\Downarrow}.a)$

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*)
$= (\forall Barbs.\ (\forall S\ x.\ x \in Barbs \wedge S{\Downarrow}{<}SWB{>}x \longrightarrow [\![S]\!]{\Downarrow}{<}TWB{>}x)$
$\wedge\ (\forall S\ x.\ x \in Barbs \wedge [\![S]\!]{\Downarrow}{<}TWB{>}x \longrightarrow S{\Downarrow}{<}SWB{>}x))$
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 ⇒ bool* **where**
enc-preserves-barb-set Barbs ≡ enc-preserves-binary-pred (λP a. a ∈ Barbs ∧ P↓.a)

abbreviation (**in** *encoding-wrt-barbs*) *enc-preserves-barbs* :: *bool* **where**
enc-preserves-barbs ≡ enc-preserves-binary-pred (λP a. P↓.a)

lemma (**in** *encoding-wrt-barbs*) *enc-preserves-barbs-and-barb-set*:
shows *enc-preserves-barbs = (∀ Barbs. enc-preserves-barb-set Barbs)*
by *blast*

abbreviation (**in** *encoding-wrt-barbs*) *enc-reflects-barb-set* :: *'barbs set ⇒ bool* **where**
enc-reflects-barb-set Barbs ≡ enc-reflects-binary-pred (λP a. a ∈ Barbs ∧ P↓.a)

abbreviation (**in** *encoding-wrt-barbs*) *enc-reflects-barbs* :: *bool* **where**
enc-reflects-barbs ≡ enc-reflects-binary-pred (λP a. P↓.a)

lemma (**in** *encoding-wrt-barbs*) *enc-reflects-barbs-and-barb-set*:
shows *enc-reflects-barbs = (∀ Barbs. enc-reflects-barb-set Barbs)*
by *blast*

abbreviation (**in** *encoding-wrt-barbs*) *enc-respects-barb-set* :: *'barbs set ⇒ bool* **where**
enc-respects-barb-set Barbs ≡ enc-preserves-barb-set Barbs ∧ enc-reflects-barb-set Barbs

abbreviation (**in** *encoding-wrt-barbs*) *enc-respects-barbs* :: *bool* **where**
enc-respects-barbs ≡ enc-preserves-barbs ∧ enc-reflects-barbs

lemma (**in** *encoding-wrt-barbs*) *enc-respects-barbs-and-barb-set*:
shows *enc-respects-barbs = (∀ Barbs. enc-respects-barb-set Barbs)*
proof −
have *(∀ Barbs. enc-respects-barb-set Barbs)*
= ((∀ Barbs. enc-preserves-barb-set Barbs)
∧ (∀ 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 × 'procT) set*
shows *enc-weakly-preserves-barb-set Barbs*
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ rel-weakly-preserves-barb-set Rel (STCalWB SWB TWB) Barbs)
using *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**
Pred=λP a. a ∈ Barbs ∧ P⇓<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*
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ rel-weakly-preserves-barbs Rel (STCalWB SWB TWB))
using *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**

Pred=λ*P a*. *P*⇓<*STCalWB SWB TWB*>*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*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-preserves-barb-set Rel* (*STCalWB SWB TWB*) *Barbs*)
using *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**
Pred=λ*P a*. *a* ∈ *Barbs* ∧ *P*↓<*STCalWB SWB TWB*>*a*] *STCalWB-hasBarbST*
by *simp*

lemma (**in** *encoding-wrt-barbs*) *enc-preserves-barbs-iff-source-target-rel*:
shows *enc-preserves-barbs*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-preserves-barbs Rel* (*STCalWB SWB TWB*))
using *enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred*[**where**
Pred=λ*P a*. *P*↓<*STCalWB SWB TWB*>*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*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-weakly-reflects-barb-set Rel* (*STCalWB SWB TWB*) *Barbs*)
using *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**
Pred=λ*P a*. *a* ∈ *Barbs* ∧ *P*⇓<*STCalWB SWB TWB*>*a*] *STCalWB-reachesBarbST*
by *simp*

lemma (**in** *encoding-wrt-barbs*) *enc-weakly-reflects-barbs-iff-source-target-rel*:
shows *enc-weakly-reflects-barbs*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-weakly-reflects-barbs Rel* (*STCalWB SWB TWB*))
using *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**
Pred=λ*P a*. *P*⇓<*STCalWB SWB TWB*>*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*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-reflects-barb-set Rel* (*STCalWB SWB TWB*) *Barbs*)
using *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**
Pred=λ*P a*. *a* ∈ *Barbs* ∧ *P*↓<*STCalWB SWB TWB*>*a*] *STCalWB-hasBarbST*
by *simp*

lemma (**in** *encoding-wrt-barbs*) *enc-reflects-barbs-iff-source-target-rel*:
shows *enc-reflects-barbs*
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-reflects-barbs Rel* (*STCalWB SWB TWB*))
using *enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred*[**where**
Pred=λ*P a*. *P*↓<*STCalWB SWB TWB*>*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 indRelL, that relates literal translations and their source terms and respects (reachability/)existence of barbs, or (3) there exists a relation, like 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*
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) *Barbs*)
    using *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**
      Pred=λ*P a*. *a* ∈ *Barbs* ∧ *P*⇓<*STCalWB SWB TWB*>*a*] *STCalWB-reachesBarbST*
  by *simp*

lemma (**in** *encoding-wrt-barbs*) *enc-weakly-respects-barbs-iff-source-target-rel*:
  shows *enc-weakly-respects-barbs*
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-weakly-respects-barbs Rel* (*STCalWB SWB TWB*))
    using *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**
      Pred=λ*P a*. *P*⇓<*STCalWB SWB TWB*>*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*
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) *Barbs*)
    using *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**
      Pred=λ*P a*. *a* ∈ *Barbs* ∧ *P*↓<*STCalWB SWB TWB*>*a*] *STCalWB-hasBarbST*
  by *simp*

lemma (**in** *encoding-wrt-barbs*) *enc-respects-barbs-iff-source-target-rel*:
  shows *enc-respects-barbs*
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-respects-barbs Rel* (*STCalWB SWB TWB*))
    using *enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR*[**where**
      Pred=λ*P a*. *P*↓<*STCalWB SWB TWB*>*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*} = (∀ *S*. *S*⇓<*SWB*>*success* ⟷ ⟦*S*⟧⇓<*TWB*>*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*}
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*})
  by (*rule enc-weakly-respects-barb-set-iff-source-target-rel*[**where** *Barbs*={*success*}])+

lemma (**in** *encoding-wrt-barbs*) *success-sensitive-iff-source-target-rel-respects-success*:
  fixes *success* :: *'barbs*
  shows *enc-respects-barb-set* {*success*}
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*})
  by (*rule enc-respects-barb-set-iff-source-target-rel*[**where** *Barbs*={*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* $(\lambda P.\ P \longmapsto ST\omega)$

lemma (**in** *encoding*) *divergence-preservation-cond*:
shows *enc-preserves-divergence* $= (\forall S.\ S \longmapsto(Source)\omega \longrightarrow [\![S]\!] \longmapsto(Target)\omega)$
by *simp*

abbreviation (**in** *encoding*) *enc-reflects-divergence* :: *bool* **where**
enc-reflects-divergence $\equiv$ *enc-reflects-pred* $(\lambda P.\ P \longmapsto ST\omega)$

lemma (**in** *encoding*) *divergence-reflection-cond*:
shows *enc-reflects-divergence* $= (\forall S.\ [\![S]\!] \longmapsto(Target)\omega \longrightarrow S \longmapsto(Source)\omega)$
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* $(\lambda P.\ P \longmapsto(Cal)\omega)$

abbreviation *rel-reflects-divergence*
$:: ('proc \times 'proc)\ set \Rightarrow 'proc\ processCalculus \Rightarrow bool$
where
rel-reflects-divergence Rel Cal $\equiv$ *rel-reflects-pred Rel* $(\lambda P.\ P \longmapsto(Cal)\omega)$

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* $(\lambda P.\ P \longmapsto ST\omega)$

lemma (**in** *encoding*) *divergence-respection-cond*:
shows *enc-respects-divergence* $= (\forall S.\ [\![S]\!] \longmapsto(Target)\omega \longleftrightarrow S \longmapsto(Source)\omega)$
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* $(\lambda P.\ P \longmapsto(Cal)\omega)$

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*
$= (\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ ([\![S]\!])) \in Rel)$
$\wedge$ *rel-preserves-divergence Rel* $(STCal\ Source\ Target))$
using *enc-preserves-pred-iff-source-target-rel-preserves-pred*(*1*)[**where** $Pred{=}\lambda P.\ P \longmapsto ST\omega$]
divergentST-STCal-divergent
by *simp*

```
lemma (in encoding) divergence-preservation-iff-source-target-rel-reflects-divergence:
  shows enc-preserves-divergence
        = (∃ Rel. (∀ S. (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
          ∧ rel-reflects-divergence Rel (STCal Source Target))
      using enc-preserves-pred-iff-source-target-rel-reflects-pred(1)[where Pred=λP. P ⟼STω]
          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
        = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
          ∧ rel-reflects-divergence Rel (STCal Source Target))
      using enc-reflects-pred-iff-source-target-rel-reflects-pred[where Pred=λP. P ⟼STω]
          divergentST-STCal-divergent
    by simp

lemma (in encoding) divergence-reflection-iff-source-target-rel-preserves-divergence:
  shows enc-reflects-divergence
        = (∃ Rel. (∀ S. (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
          ∧ rel-preserves-divergence Rel (STCal Source Target))
      using enc-reflects-pred-iff-source-target-rel-preserves-pred[where Pred=λP. P ⟼STω]
          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 = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
        ∧ rel-respects-divergence Rel (STCal Source Target))
  and   enc-respects-divergence = (∃ Rel.
        (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
        ∧ rel-respects-divergence Rel (STCal Source Target))
proof −
  show enc-respects-divergence = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
      using enc-respects-pred-iff-source-target-rel-respects-pred-encR[where Pred=λP. P ⟼STω]
          divergentST-STCal-divergent
    by simp
next
  show enc-respects-divergence = (∃ Rel.
       (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
      using enc-respects-pred-iff-source-target-rel-respects-pred-encRL[where Pred=λP. P ⟼STω]
          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 $\equiv$
$\forall S\ S'.\ S \longmapsto Source{*}\ S' \longrightarrow (\exists T.\ [\![S]\!] \longmapsto Target{*}\ T \wedge ([\![S']\!],\ T) \in TRel)$

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 $\equiv$
$\forall S\ S'.\ S \longmapsto Source\ S' \longrightarrow (\exists T.\ [\![S]\!] \longmapsto Target\ T \wedge ([\![S']\!],\ T) \in TRel)$

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 $\equiv$
$\forall S\ T.\ [\![S]\!] \longmapsto Target{*}\ T \longrightarrow (\exists S'\ T'.\ S \longmapsto Source{*}\ S' \wedge T \longmapsto Target{*}\ T' \wedge ([\![S']\!],\ T') \in TRel)$

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 a 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 $\equiv \forall S\ T.\ [\![S]\!] \longmapsto Target{*}\ T \longrightarrow (\exists S'.\ S \longmapsto Source{*}\ S' \wedge ([\![S']\!],\ T) \in TRel)$

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 $\equiv$
$\forall S\ T.\ [\![S]\!] \longmapsto Target\ T \longrightarrow (\exists S'.\ S \longmapsto Source\ S' \wedge ([\![S']\!],\ T) \in TRel)$

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* $\{(T1, T2). \; True\}$
    and *weakly-operational-sound* $\{(T1, T2). \; True\}$
    and *operational-sound* $\{(T1, T2). \; True\}$
      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* $\{(a, b). \; True\}$ *Cal*
      using *steps-refl*[**where** *Cal*=*Cal*]
    by *blast*

lemma (**in** *encoding*) *operational-correspondence-modulo-some-target-relation*:
  shows $\exists$ *TRel*. *weakly-operational-corresponding TRel*
    and $\exists$ *TRel*. *operational-corresponding TRel*
    and $\exists$ *TRel*. *weakly-operational-corresponding TRel* $\wedge$ *weak-reduction-bisimulation TRel Target*
    and $\exists$ *TRel*. *operational-corresponding TRel* $\wedge$ *weak-reduction-bisimulation TRel 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*. *strongly-operational-corresponding TRel*)
      $= (\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T))$
    and ($\exists$ *TRel*. *strongly-operational-corresponding TRel*
        $\wedge$ *weak-reduction-bisimulation TRel Target*)
      $= (\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T))$
proof −
  have *A1*: $\exists$ *TRel*. *strongly-operational-corresponding TRel*
        $\Longrightarrow \forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T)$
    by *blast*
  moreover have *A2*: $\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T)$
        $\Longrightarrow \exists$ *TRel*. *strongly-operational-corresponding TRel*
          $\wedge$ *weak-reduction-bisimulation TRel Target*
  proof −
    assume $\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T)$
    hence *strongly-operational-corresponding* $\{(T1, T2). \; True\}$
      by *simp*
    thus $\exists$ *TRel*. *strongly-operational-corresponding TRel*
        $\wedge$ *weak-reduction-bisimulation TRel Target*
        using *all-relation-is-weak-reduction-bisimulation*[**where** *Cal*=*Target*]
      by *blast*
  qed
  ultimately show ($\exists$ *TRel*. *strongly-operational-corresponding TRel*
          $\wedge$ *weak-reduction-bisimulation TRel Target*)
        $= (\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T))$
    by *blast*
  from *A1 A2* **show** ($\exists$ *TRel*. *strongly-operational-corresponding TRel*)
        $= (\forall S. \; (\exists S'. \; S \longmapsto Source \; S') \longleftrightarrow (\exists T. \; [\![S]\!] \longmapsto Target \; T))$
    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* ($[\![S]\!]$)) $\in$ *Rel*
and *A2*: $\forall S\ T.$ (*SourceTerm S*, *TargetTerm T*) $\in$ *Rel* $\longrightarrow$ ($[\![S]\!]$, *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* ($[\![S]\!]$)) $\in$ *Rel*
by *simp*
moreover assume $S \longmapsto Source{*}\ S'$
hence *SourceTerm S* $\longmapsto$(*STCal Source Target*)$*$ (*SourceTerm* S')
by (*simp add*: *STCal-steps*(*1*))
ultimately obtain Q' **where** *A5*: *TargetTerm* ($[\![S]\!]$) $\longmapsto$(*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*: $[\![S]\!] \longmapsto Target{*}\ T$
by (*auto simp add*: *STCal-steps*(*2*))
from *A2 A6 A7* **have** ($[\![S']\!]$, *T*) $\in$ $TRel^*$
by *simp*
with *A8* **show** $\exists T.\ [\![S]\!] \longmapsto Target{*}\ T \wedge ([\![S']\!], T) \in TRel^*$
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** $P \longmapsto$(*STCal Source Target*)$*$ P'
thus $\exists Q'.\ Q \longmapsto$(*STCal Source Target*)$*$ $Q' \wedge P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
proof (*induct arbitrary*: P')
case (*encR S*)
assume *SourceTerm S* $\longmapsto$(*STCal Source Target*)$*$ P'
from *this* **obtain** S' **where** *A1*: $S' \in S\ P'$ **and** *A2*: $S \longmapsto Source{*}\ S'$
by (*auto simp add*: *STCal-steps*(*1*))
from *oc A2* **obtain** *T* **where** *A3*: $[\![S]\!] \longmapsto Target{*}\ T$ **and** *A4*: ($[\![S']\!]$, *T*) $\in$ $TRel^*$
by *blast*
from *A3* **have** *TargetTerm* ($[\![S]\!]$) $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm T*)
by (*simp add*: *STCal-steps*(*2*))
moreover have $P' \lesssim[\![\cdot]\!]RT{<}TRel{>}$ *TargetTerm T*
proof $-$
from *A4* **have** $[\![S']\!] = T \vee ([\![S']\!], T) \in TRel^+$
using *rtrancl-eq-or-trancl*[*of* $[\![S']\!]$ *T TRel*]
by *blast*
moreover from *A1* **have** *A5*: $P' \lesssim[\![\cdot]\!]RT{<}TRel{>}$ *TargetTerm* ($[\![S']\!]$)
by (*simp add*: *indRelRTPO.encR*)

```
    hence ⟦S'⟧ = T ⟹ P' ≲⟦·⟧RT<TRel> TargetTerm T
      by simp
    moreover have (⟦S'⟧, T) ∈ TRel⁺ ⟹ P' ≲⟦·⟧RT<TRel> TargetTerm T
    proof –
      assume (⟦S'⟧, T) ∈ TRel⁺
      hence TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm T
      proof induct
        fix T
        assume (⟦S'⟧, T) ∈ TRel
        thus TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm T
          by (rule indRelRTPO.target)
      next
        case (step TQ TR)
        assume TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm TQ
        moreover assume (TQ, TR) ∈ TRel
        hence TargetTerm TQ ≲⟦·⟧RT<TRel> TargetTerm TR
          by (rule indRelRTPO.target)
        ultimately show TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm TR
          by (rule indRelRTPO.trans)
      qed
      with A5 show P' ≲⟦·⟧RT<TRel> TargetTerm T
        by (rule indRelRTPO.trans)
    qed
    ultimately show P' ≲⟦·⟧RT<TRel> TargetTerm T
      by blast
  qed
  ultimately
  show ∃ Q'. TargetTerm (⟦S⟧) ⟼(STCal Source Target)∗ Q' ∧ P' ≲⟦·⟧RT<TRel> Q'
    by blast
next
  case (source S)
  then obtain S' where B1: S' ∈S P'
    by (auto simp add: STCal-steps(1))
  hence P' ≲⟦·⟧RT<TRel> P'
    by (simp add: indRelRTPO.source)
  with source show ∃ Q'. SourceTerm S ⟼(STCal Source Target)∗ Q' ∧ P' ≲⟦·⟧RT<TRel> Q'
    by blast
next
  case (target T1 T2)
  assume TargetTerm T1 ⟼(STCal Source Target)∗ P'
  from this obtain T1' where C1: T1' ∈T P' and C2: T1 ⟼Target∗ T1'
    by (auto simp add: STCal-steps(2))
  assume (T1, T2) ∈ TRel
  hence (T1, T2) ∈ TRel⁺
    by simp
  with C2 sim obtain T2' where C3: T2 ⟼Target∗ T2'
                         and C4: (T1', T2') ∈ TRel⁺
    by blast
  from C3 have TargetTerm T2 ⟼(STCal Source Target)∗ (TargetTerm T2')
    by (simp add: STCal-steps(2))
  moreover from C4 have TargetTerm T1' ≲⟦·⟧RT<TRel> TargetTerm T2'
  proof induct
    fix T2'
    assume (T1', T2') ∈ TRel
    thus TargetTerm T1' ≲⟦·⟧RT<TRel> TargetTerm T2'
      by (rule indRelRTPO.target)
  next
    case (step TQ TR)
    assume TargetTerm T1' ≲⟦·⟧RT<TRel> TargetTerm TQ
    moreover assume (TQ, TR) ∈ TRel
    hence TargetTerm TQ ≲⟦·⟧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 $\exists\, Q'.\ TargetTerm\ T2 \longmapsto(STCal\ Source\ Target){*}\ Q' \wedge P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
by *blast*
next
case (*trans P Q R*)
assume $P \longmapsto(STCal\ Source\ Target){*}\ P'$
and $\bigwedge P'.\ P \longmapsto(STCal\ Source\ Target){*}\ P'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto(STCal\ Source\ Target){*}\ Q' \wedge P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
from *this* **obtain** Q' **where** *D1*: $Q \longmapsto(STCal\ Source\ Target){*}\ Q'$
and *D2*: $P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ Q'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(STCal\ Source\ Target){*}\ Q'$
$\Longrightarrow \exists\, R'.\ R \longmapsto(STCal\ Source\ Target){*}\ R' \wedge Q' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R'$
with *D1* **obtain** R' **where** *D3*: $R \longmapsto(STCal\ Source\ Target){*}\ R'$
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** $\exists\, R'.\ R \longmapsto(STCal\ Source\ Target){*}\ R' \wedge P' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ R'$
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*) (*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* ($TRel^+$) *Target*
using *indRelRTPO-impl-TRel-is-weak-reduction-simulation*[**where** *TRel*=*TRel*]
by *simp*
ultimately show *operational-complete* ($TRel^*$)
$\wedge$ *weak-reduction-simulation* ($TRel^+$) *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* ($TRel^+$) *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* (*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 ∀ *S T*. *SourceTerm S* ≲⟦·⟧*RT*<*TRel*> *TargetTerm T* ⟶ (⟦*S*⟧, *T*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*⁺)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel**)
∧ *weak-reduction-simulation Rel* (*STCal Source Target*)
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*⁺)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel**)
∧ *weak-reduction-simulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and *A3*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*⁺
and *A4*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel**
and *A5*: *weak-reduction-simulation Rel* (*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**)
∧ *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 *Rel* :: ((′*procS*, ′*procT*) *Proc* × (′*procS*, ′*procT*) *Proc*) *set*
and *TRel* :: (′*procT* × ′*procT*) *set*
assumes *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel**
and *A3*: *strong-reduction-simulation Rel* (*STCal Source Target*)
shows *strongly-operational-complete* (*TRel**)
proof *clarify*
fix *S S*′
from *A1* **have** (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
by *simp*
moreover assume *S* ⟼*Source S*′
hence *SourceTerm S* ⟼(*STCal Source Target*) (*SourceTerm S*′)
by (*simp add*: *STCal-step*(*1*))
ultimately obtain *Q*′ **where** *A5*: *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*) *Q*′
and *A6*: (*SourceTerm S*′, *Q*′) ∈ *Rel*
using *A3*
by *blast*

```
  from A5 obtain T where A7: T ∈T Q' and A8: ⟦S⟧ ⟼Target T
    by (auto simp add: STCal-step(2))
  from A2 A6 A7 have (⟦S'⟧, T) ∈ TRel^*
    by simp
  with A8 show ∃ T. ⟦S⟧ ⟼Target T ∧ (⟦S'⟧, T) ∈ TRel^*
    by blast
qed

lemma (in encoding) SOCom-iff-indRelRTPO-is-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  shows (strongly-operational-complete (TRel^*)
         ∧ strong-reduction-simulation (TRel^+) Target)
         = strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
proof (rule iffI, erule conjE)
  assume soc: strongly-operational-complete (TRel^*)
     and sim: strong-reduction-simulation (TRel^+) Target
  show strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
  proof clarify
    fix P Q P'
    assume P ≲⟦·⟧RT<TRel> Q and P ⟼(STCal Source Target) P'
    thus ∃ Q'. Q ⟼(STCal Source Target) Q' ∧ P' ≲⟦·⟧RT<TRel> Q'
    proof (induct arbitrary: P')
      case (encR S)
      assume SourceTerm S ⟼(STCal Source Target) P'
      from this obtain S' where A1: S' ∈S P' and A2: S ⟼Source S'
        by (auto simp add: STCal-step(1))
      from soc A2 obtain T where A3: ⟦S⟧ ⟼Target T and A4: (⟦S'⟧, T) ∈ TRel^*
        by blast
      from A3 have TargetTerm (⟦S⟧) ⟼(STCal Source Target) (TargetTerm T)
        by (simp add: STCal-step(2))
      moreover have P' ≲⟦·⟧RT<TRel> TargetTerm T
      proof −
        from A4 have ⟦S'⟧ = T ∨ (⟦S'⟧, T) ∈ TRel^+
            using rtrancl-eq-or-trancl[of ⟦S'⟧ T TRel]
          by blast
        moreover from A1 have A5: P' ≲⟦·⟧RT<TRel> TargetTerm (⟦S'⟧)
          by (simp add: indRelRTPO.encR)
        hence ⟦S'⟧ = T ⟹ P' ≲⟦·⟧RT<TRel> TargetTerm T
          by simp
        moreover have (⟦S'⟧, T) ∈ TRel^+ ⟹ P' ≲⟦·⟧RT<TRel> TargetTerm T
        proof −
          assume (⟦S'⟧, T) ∈ TRel^+
          hence TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm T
          proof induct
            fix TQ
            assume (⟦S'⟧, TQ) ∈ TRel
            thus TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm TQ
              by (rule indRelRTPO.target)
          next
            case (step TQ TR)
            assume TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm TQ
            moreover assume (TQ, TR) ∈ TRel
            hence TargetTerm TQ ≲⟦·⟧RT<TRel> TargetTerm TR
              by (rule indRelRTPO.target)
            ultimately show TargetTerm (⟦S'⟧) ≲⟦·⟧RT<TRel> TargetTerm TR
              by (rule indRelRTPO.trans)
          qed
          with A5 show P' ≲⟦·⟧RT<TRel> TargetTerm T
            by (rule indRelRTPO.trans)
        qed
        ultimately show P' ≲⟦·⟧RT<TRel> TargetTerm T
```

by *blast*
qed
ultimately
show $\exists\, Q'.\ TargetTerm\ (\llbracket S\rrbracket) \longmapsto(STCal\ Source\ Target)\ Q' \wedge P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
by *blast*
next
case $(source\ S)$
then obtain S' **where** $B1$: $S' \in S\ P'$
by $(auto\ simp\ add{:}\ STCal\text{-}step(1))$
hence $P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ P'$
by $(simp\ add{:}\ indRelRTPO.source)$
with *source* **show** $\exists\, Q'.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)\ Q' \wedge P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
by *blast*
next
case $(target\ T1\ T2)$
assume $TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)\ P'$
from *this* **obtain** $T1'$ **where** $C1$: $T1' \in T\ P'$ **and** $C2$: $T1 \longmapsto Target\ T1'$
by $(auto\ simp\ add{:}\ STCal\text{-}step(2))$
assume $(T1,\ T2) \in TRel$
hence $(T1,\ T2) \in TRel^+$
by *simp*
with *C2 sim* **obtain** $T2'$ **where** $C3$: $T2 \longmapsto Target\ T2'$ **and** $C4$: $(T1',\ T2') \in TRel^+$
by *blast*
from *C3* **have** $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)\ (TargetTerm\ T2')$
by $(simp\ add{:}\ STCal\text{-}step(2))$
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 $TargetTerm\ T1' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ TargetTerm\ TR$
by $(rule\ indRelRTPO.trans)$
qed
with *C1* **have** $P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ TargetTerm\ T2'$
by *simp*
ultimately show $\exists\, Q'.\ TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)\ Q' \wedge P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
by *blast*
next
case $(trans\ P\ Q\ R)$
assume $P \longmapsto(STCal\ Source\ Target)\ P'$
and $\bigwedge P'.\ P \longmapsto(STCal\ Source\ Target)\ P'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto(STCal\ Source\ Target)\ Q' \wedge P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
from *this* **obtain** Q' **where** $D1$: $Q \longmapsto(STCal\ Source\ Target)\ Q'$
and $D2$: $P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(STCal\ Source\ Target)\ Q'$
$\Longrightarrow \exists\, R'.\ R \longmapsto(STCal\ Source\ Target)\ R' \wedge Q' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ R'$
with *D1* **obtain** R' **where** $D3$: $R \longmapsto(STCal\ Source\ Target)\ R'$
and $D4$: $Q' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ R'$
by *blast*
from *D2 D4* **have** $P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ R'$
by $(rule\ indRelRTPO.trans)$
with *D3* **show** $\exists\, R'.\ R \longmapsto(STCal\ Source\ Target)\ R' \wedge P' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ R'$
by *blast*

    qed
  qed
next
  have $\forall S.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ ([\![S]\!])$
    by (*simp add*: *indRelRTPO.encR*)
  moreover have $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$
      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* × *'procT*) *set*
  shows (*strongly-operational-complete* (TRel^*)
        $\wedge$ *strong-reduction-simulation* (TRel^+) *Target*)
        $= (\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
          $\wedge\ (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
          $\wedge\ (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
          $\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
          $\wedge$ *strong-reduction-simulation Rel* (*STCal Source Target*))
proof (*rule iffI*, *erule conjE*)
  have $\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{indRelRTPO}\ \mathit{TRel}$
    by (*simp add*: *indRelRTPO.encR*)
  moreover have $\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T2$
    by (*simp add*: *indRelRTPO.target*)
  moreover have $\forall T1\ T2.\ \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T2 \longrightarrow (T1,\ T2) \in \mathit{TRel}^+$
      using *indRelRTPO-to-TRel*(*4*)[**where** *TRel*=*TRel*]
    by *simp*
  moreover have $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$
      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 $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
                $\wedge\ (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
                $\wedge\ (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
                $\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
                $\wedge$ *strong-reduction-simulation Rel* (*STCal Source Target*)
    by *blast*
next
  assume $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
        $\wedge\ (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
        $\wedge\ (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
        $\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
        $\wedge$ *strong-reduction-simulation Rel* (*STCal Source Target*)
  from *this* **obtain** *Rel* **where** *A1*: $\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel}$
    and *A2*: $\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}$
    and *A3*: $\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+$
    and *A4*: $\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$

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* ($[\![S]\!]$), *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* $\longrightarrow$ ($[\![S]\!]$, *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* $\longrightarrow$ ($[\![S]\!]$, *T*) $\in$ $TRel^*$
by *blast*
ultimately show *?thesis*
by *blast*
qed

lemma (**in** *encoding*) *SOCom-modulo-TRel-iff-strong-reduction-simulation*:
shows ($\exists$ *TRel*. *strongly-operational-complete* ($TRel^*$)
$\wedge$ *strong-reduction-simulation* ($TRel^+$) *Target*)
= ($\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ ($\forall$ *S T*. (*SourceTerm S*, *TargetTerm T*) $\in$ *Rel* $\longrightarrow$ (*TargetTerm* ($[\![S]\!]$), *TargetTerm T*) $\in$ $Rel^=$)
$\wedge$ *strong-reduction-simulation Rel* (*STCal Source Target*))
proof (*rule iffI*)
assume $\exists$ *TRel*. *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* ($[\![S]\!]$)
by (*simp add*: *indRelRTPO.encR*)
moreover have $\forall$ *S T*. *SourceTerm S* $\lesssim[\![\cdot]\!]RT$<*TRel*> *TargetTerm T*
$\longrightarrow$ (*TargetTerm* ($[\![S]\!]$), *TargetTerm T*) $\in$ $(indRelRTPO\ TRel)^=$
using *indRelRTPO-relates-source-target*[**where** *TRel=TRel*]
by *simp*
ultimately show $\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ ($\forall$ *S T*. (*SourceTerm S*, *TargetTerm T*) $\in$ *Rel*
$\longrightarrow$ (*TargetTerm* ($[\![S]\!]$), *TargetTerm T*) $\in$ $Rel^=$)
$\wedge$ *strong-reduction-simulation Rel* (*STCal Source Target*)
by *blast*

next
  assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^{=}$)
  ∧ *strong-reduction-simulation Rel* (*STCal Source Target*)
  from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
  and *A2*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
  ⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^{=}$)
  and *A3*: *strong-reduction-simulation Rel* (*STCal Source Target*)
  by *blast*
  from *A2* **obtain** *TRel* **where** ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
  and ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^{+}$
  and ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^{*}$
  using *target-relation-from-source-target-relation*[**where** *Rel*=*Rel*]
  by *blast*
  with *A1 A3* **have** *strongly-operational-complete* ($TRel^{*}$)
  ∧ *strong-reduction-simulation* ($TRel^{+}$) *Target*
  using *SOCom-iff-strong-reduction-simulation*[**where** *TRel*=*TRel*]
  by *blast*
  thus ∃ *TRel*. *strongly-operational-complete* ($TRel^{*}$)
  ∧ *strong-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* × ('*procS*, '*procT*) *Proc*) *set*
  and *TRel* :: ('*procT* × '*procT*) *set*
  assumes *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
  and *A2*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^{*}$
  and *A3*: *weak-reduction-contrasimulation* (Rel^{-1}) (*STCal Source Target*)
  shows *weakly-operational-sound* ($TRel^{*}$)
proof *clarify*
  fix *S T*
  from *A1* **have** (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ Rel^{-1}
  by *simp*
  moreover assume ⟦*S*⟧ ⟼*Target*∗ *T*
  hence *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*)∗ (*TargetTerm T*)
  by (*simp add*: *STCal-steps*(*2*))
  ultimately obtain *Q'* **where** *A5*: *SourceTerm S* ⟼(*STCal Source Target*)∗ *Q'*
  and *A6*: (*Q'*, *TargetTerm T*) ∈ Rel^{-1}
  using *A3*
  by *blast*
  from *A5* **obtain** *S'* **where** *A7*: *S'* ∈*S Q'* **and** *A8*: *S* ⟼*Source*∗ *S'*
  by (*auto simp add*: *STCal-steps*(*1*))
  have *Q'* ⟼(*STCal Source Target*)∗ *Q'*
  by (*simp add*: *steps-refl*)
  with *A6 A3* **obtain** *P''* **where** *A9*: *TargetTerm T* ⟼(*STCal Source Target*)∗ *P''*
  and *A10*: (*P''*, *Q'*) ∈ Rel^{-1}
  by *blast*
  from *A9* **obtain** *T'* **where** *A11*: *T'* ∈*T P''* **and** *A12*: *T* ⟼*Target*∗ *T'*
  by (*auto simp add*: *STCal-steps*(*2*))
  from *A10* **have** (*Q'*, *P''*) ∈ *Rel*
  by *induct*
  with *A2 A7 A11* **have** (⟦*S'*⟧, *T'*) ∈ $TRel^{*}$
  by *simp*
  with *A8 A12* **show** ∃ *S' T'*. *S* ⟼*Source*∗ *S'* ∧ *T* ⟼*Target*∗ *T'* ∧ (⟦*S'*⟧, *T'*) ∈ $TRel^{*}$
  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 $A3$: *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 $Q \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ P$ **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 $SourceTerm\ S' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ P'$
proof −
from $A4$ **have** $\llbracket S' \rrbracket = T \vee (\llbracket S' \rrbracket,\ T) \in TRel^+$

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** $[\![S']\!] = T \Longrightarrow$ *SourceTerm* $S' \lesssim[\![\cdot]\!]RT<TRel>$ P'
by *simp*
moreover have ($[\![S']\!]$, T) $\in$ $TRel^+$ $\Longrightarrow$ *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* $\longmapsto$(*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* $\longmapsto$(*STCal Source Target*)$*$ Q' $\wedge$ (P', Q') $\in$ (*indRelRTPO TRel*)$^{-1}$
by *blast*
next
case (*target T1 T2*)
assume *TargetTerm T2* $\longmapsto$(*STCal Source Target*)$*$ P'
from *this* **obtain** $T2'$ **where** *C1*: $T2' \in T\ P'$ **and** *C2*: *T2* $\longmapsto$*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* $\longmapsto$*Target*$*$ $T1'$ **and** *C4*: ($T2'$, $T1'$) $\in$ ($TRel^+$)$^{-1}$
by *blast*
from *C3* **have** *TargetTerm T1* $\longmapsto$(*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* $\longmapsto$(*STCal Source Target*)$*$ Q' $\wedge$ (P', Q') $\in$ (*indRelRTPO TRel*)$^{-1}$
by *blast*
next
case (*trans P Q R* R')
assume *R* $\longmapsto$(*STCal Source Target*)$*$ R'
and $\bigwedge R'$. *R* $\longmapsto$(*STCal Source Target*)$*$ R'
$\Longrightarrow \exists\, Q'$. *Q* $\longmapsto$(*STCal Source Target*)$*$ Q' $\wedge$ (R', Q') $\in$ (*indRelRTPO TRel*)$^{-1}$
from *this* **obtain** Q' **where** *D1*: *Q* $\longmapsto$(*STCal Source Target*)$*$ Q'
and *D2*: (R', Q') $\in$ (*indRelRTPO TRel*)$^{-1}$

by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(\mathit{STCal\ Source\ Target}){*}\ Q'$
$\Longrightarrow \exists P'.\ P \longmapsto(\mathit{STCal\ Source\ Target}){*}\ P' \wedge (Q', P') \in (\mathit{indRelRTPO\ TRel})^{-1}$
with *D1* **obtain** P' **where** *D3*: $P \longmapsto(\mathit{STCal\ Source\ Target}){*}\ P'$
and *D4*: $(Q', P') \in (\mathit{indRelRTPO\ TRel})^{-1}$
by *blast*
from *D4 D2* **have** $(R', P') \in (\mathit{indRelRTPO\ TRel})^{-1}$
by (*simp add*: *indRelRTPO.trans*[**where** $P=P'$ **and** $Q=Q'$ **and** $R=R'$])
with *D3* **show** $\exists P'.\ P \longmapsto(\mathit{STCal\ Source\ Target}){*}\ P' \wedge (R', P') \in (\mathit{indRelRTPO\ TRel})^{-1}$
by *blast*
qed
qed
next
have $\forall S.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ ([\![S]\!])$
by (*simp add*: *indRelRTPO.encR*)
moreover have $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!], T) \in \mathit{TRel}^*$
using *indRelRTPO-to-TRel*(*2*)[**where** *TRel*=*TRel*] *trans-closure-of-TRel-refl-cond*
by *simp*
moreover
assume *sim*: *weak-reduction-simulation* $((\mathit{indRelRTPO\ TRel})^{-1})$ (*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* $((\mathit{TRel}^+)^{-1})$ *Target*
using *indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev*[**where** *TRel*=*TRel*]
by *simp*
ultimately show *operational-sound* $(\mathit{TRel}^*) \wedge$ *weak-reduction-simulation* $((\mathit{TRel}^+)^{-1})$ *Target*
by *simp*
qed

lemma (**in** *encoding*) *OSou-iff-weak-reduction-simulation*:
fixes *TRel* :: $('\mathit{procT} \times '\mathit{procT})\ \mathit{set}$
shows (*operational-sound* (TRel^*)
$\wedge$ *weak-reduction-simulation* $((\mathit{TRel}^+)^{-1})$ *Target*)
$= (\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S, \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (T1, T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1, \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1, \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1, T2) \in \mathit{TRel}^+)$
$\wedge (\forall S\ T.\ (\mathit{SourceTerm}\ S, \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!], T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-simulation* (Rel^{-1}) (*STCal Source Target*))
proof (*rule iffI*, *erule conjE*)
have $\forall S.\ (\mathit{SourceTerm}\ S, \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{indRelRTPO\ TRel}$
by (*simp add*: *indRelRTPO.encR*)
moreover have $\forall T1\ T2.\ (T1, T2) \in \mathit{TRel} \longrightarrow \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T2$
by (*simp add*: *indRelRTPO.target*)
moreover have $\forall T1\ T2.\ \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T2 \longrightarrow (T1, T2) \in \mathit{TRel}^+$
using *indRelRTPO-to-TRel*(*4*)[**where** *TRel*=*TRel*]
by *simp*
moreover have $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!], T) \in \mathit{TRel}^*$
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* $((\mathit{TRel}^+)^{-1})$ *Target*
hence *weak-reduction-simulation* $((\mathit{indRelRTPO\ TRel})^{-1})$ (*STCal Source Target*)
using *OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation*[**where** *TRel*=*TRel*]
by *simp*
ultimately show $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S, \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (T1, T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1, \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1, \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1, T2) \in \mathit{TRel}^+)$
$\wedge (\forall S\ T.\ (\mathit{SourceTerm}\ S, \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!], T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
by *blast*

next
  assume $\exists$ *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*)
  from *this* **obtain** *Rel* **where** *A1*: $\forall$ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) $\in$ *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* $\longrightarrow$ ([[*S*]], *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* ([[*S*]])) $\in$ *Rel*
      and *A2*: $\forall$ *S T*. (*SourceTerm S*, *TargetTerm T*) $\in$ *Rel* $\longrightarrow$ ([[*S*]], *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* ([[*S*]]), *SourceTerm S*) $\in$ *Rel*$^{-1}$
    by *simp*
  moreover assume [[*S*]] $\longmapsto$*Target T*
  hence *TargetTerm* ([[*S*]]) $\longmapsto$(*STCal Source Target*) (*TargetTerm T*)
    by (*simp add*: *STCal-step*(*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-step*(*1*))
  from *A6* **have** (*Q'*, *TargetTerm T*) $\in$ *Rel*
    by *induct*
  with *A2 A7* **have** ([[*S'*]], *T*) $\in$ *TRel**
    by *simp*
  with *A8* **show** $\exists$ *S'*. *S* $\longmapsto$*Source S'* $\wedge$ ([[*S'*]], *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 $P \longmapsto(STCal\ Source\ Target)\ P'$
ultimately
show $\exists\, Q'.\ Q \longmapsto(STCal\ Source\ Target)\ Q' \wedge (P',\ Q') \in (indRelRTPO\ TRel)^{-1}$
proof (*induct arbitrary*: P')
case (*encR S*)
assume $TargetTerm\ ([\![S]\!]) \longmapsto(STCal\ Source\ Target)\ P'$
from *this* **obtain** T **where** *A1*: $T \in T\ P'$ **and** *A2*: $[\![S]\!] \longmapsto Target\ T$
by (*auto simp add*: *STCal-step*(*2*))
from *os A2* **obtain** S' **where** *A3*: $S \longmapsto Source\ S'$ **and** *A4*: $([\![S']\!],\ T) \in TRel^*$
by *blast*
from *A3* **have** $SourceTerm\ S \longmapsto(STCal\ Source\ Target)\ (SourceTerm\ S')$
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** $[\![S']\!] = T \Longrightarrow SourceTerm\ S' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ P'$
by *simp*
moreover have $([\![S']\!],\ T) \in TRel^+ \Longrightarrow SourceTerm\ S' \lesssim[\![\cdot]\!]RT{<}TRel{>}\ P'$
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 $\exists\, Q'.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)\ Q' \wedge (P',\ Q') \in (indRelRTPO\ TRel)^{-1}$
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 $\exists\, Q'.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)\ Q' \wedge (P',\ Q') \in (indRelRTPO\ TRel)^{-1}$
by *blast*
next
case (*target T1 T2*)
assume $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)\ P'$
from *this* **obtain** $T2'$ **where** *C1*: $T2' \in T\ P'$ **and** *C2*: $T2 \longmapsto Target\ T2'$
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*: $T1 \longmapsto Target\ T1'$ **and** *C4*: $(T2',\ T1') \in (TRel^+)^{-1}$
by *blast*
from *C3* **have** $TargetTerm\ T1 \longmapsto (STCal\ Source\ Target)\ (TargetTerm\ T1')$
by (*simp add*: *STCal-step*(*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 $\exists\, Q'.\ TargetTerm\ T1 \longmapsto (STCal\ Source\ Target)\ Q' \wedge (P',\ Q') \in (indRelRTPO\ TRel)^{-1}$
by *blast*
next
case (*trans P Q R R'*)
assume $R \longmapsto (STCal\ Source\ Target)\ R'$
and $\bigwedge R'.\ R \longmapsto (STCal\ Source\ Target)\ R'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto (STCal\ Source\ Target)\ Q' \wedge (R',\ Q') \in (indRelRTPO\ TRel)^{-1}$
from *this* **obtain** Q' **where** *D1*: $Q \longmapsto (STCal\ Source\ Target)\ Q'$
and *D2*: $(R',\ Q') \in (indRelRTPO\ TRel)^{-1}$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto (STCal\ Source\ Target)\ Q'$
$\Longrightarrow \exists\, P'.\ P \longmapsto (STCal\ Source\ Target)\ P' \wedge (Q',\ P') \in (indRelRTPO\ TRel)^{-1}$
with *D1* **obtain** P' **where** *D3*: $P \longmapsto (STCal\ Source\ Target)\ P'$
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** $\exists\, P'.\ P \longmapsto (STCal\ Source\ Target)\ P' \wedge (R',\ P') \in (indRelRTPO\ TRel)^{-1}$
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*: *strong-reduction-simulation* $((indRelRTPO\ TRel)^{-1})$ $(STCal\ Source\ Target)$
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* $((TRel^+)^{-1})$ *Target*
using *indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev*[**where** $TRel{=}TRel$]
by *simp*
ultimately
show *strongly-operational-sound* $(TRel^*) \wedge$ *strong-reduction-simulation* $((TRel^+)^{-1})$ *Target*
by *simp*
qed

lemma (**in** *encoding*) *SOSou-iff-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
shows (*strongly-operational-sound* $(TRel^*) \wedge$ *strong-reduction-simulation* $((TRel^+)^{-1})$ *Target*)
$= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ ([\![S]\!])) \in Rel)$
$\wedge\ (\forall\, T1\ T2.\ (T1,\ T2) \in TRel \longrightarrow (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel)$
$\wedge\ (\forall\, T1\ T2.\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel \longrightarrow (T1,\ T2) \in TRel^+)$
$\wedge\ (\forall\, S\ T.\ (SourceTerm\ S,\ TargetTerm\ T) \in Rel \longrightarrow ([\![S]\!],\ T) \in TRel^*)$
$\wedge$ *strong-reduction-simulation* (Rel^{-1}) $(STCal\ Source\ Target))$
proof (*rule iffI*, *erule conjE*)

have ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelRTPO TRel
by (simp add: indRelRTPO.encR)
moreover have ∀ T1 T2. (T1, T2) ∈ TRel ⟶ TargetTerm T1 ≲⟦·⟧RT<TRel> TargetTerm T2
by (simp add: indRelRTPO.target)
moreover have ∀ T1 T2. TargetTerm T1 ≲⟦·⟧RT<TRel> TargetTerm T2 ⟶ (T1, T2) ∈ TRel⁺
using indRelRTPO-to-TRel(4)[**where** TRel=TRel]
by simp
moreover have ∀ S T. SourceTerm S ≲⟦·⟧RT<TRel> TargetTerm T ⟶ (⟦S⟧, T) ∈ TRel*
using indRelRTPO-to-TRel(2)[**where** TRel=TRel] trans-closure-of-TRel-refl-cond
by simp
moreover assume strongly-operational-sound (TRel*)
and strong-reduction-simulation ((TRel⁺)⁻¹) Target
hence 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
∧ strong-reduction-simulation (Rel⁻¹) (STCal Source Target)
by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
∧ strong-reduction-simulation (Rel⁻¹) (STCal Source Target)
from this **obtain** Rel **where** A1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
and A2: ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
and A3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
and A4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*
and A5: strong-reduction-simulation (Rel⁻¹) (STCal Source Target)
by blast
from A1 A4 A5 **have** strongly-operational-sound (TRel*)
using strong-reduction-simulation-impl-SOSou[**where** Rel=Rel **and** TRel=TRel]
by simp
moreover from A2 A3 A5 **have** strong-reduction-simulation ((TRel⁺)⁻¹) Target
using rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev[**where** Rel=Rel **and**
TRel=TRel]
by simp
ultimately
show strongly-operational-sound (TRel*) ∧ strong-reduction-simulation ((TRel⁺)⁻¹) Target
by simp
qed

lemma (**in** encoding) SOSou-modulo-TRel-iff-strong-reduction-simulation:
shows (∃ TRel. strongly-operational-sound (TRel*)
∧ strong-reduction-simulation ((TRel⁺)⁻¹) Target)
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (TargetTerm (⟦S⟧), TargetTerm T) ∈ Rel⁼)
∧ strong-reduction-simulation (Rel⁻¹) (STCal Source Target))
proof (rule iffI)
assume ∃ TRel. strongly-operational-sound (TRel*)
∧ strong-reduction-simulation ((TRel⁺)⁻¹) Target
from this **obtain** TRel **where** strongly-operational-sound (TRel*)
and strong-reduction-simulation ((TRel⁺)⁻¹) Target
by blast
hence strong-reduction-simulation ((indRelRTPO TRel)⁻¹) (STCal Source Target)
using SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation[**where** TRel=TRel]
by simp
moreover have ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelRTPO TRel

by (*simp add*: *indRelRTPO.encR*)
moreover have ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *indRelRTPO TRel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $(indRelRTPO\ TRel)^=$
using *indRelRTPO-relates-source-target*[**where** *TRel*=*TRel*]
by *simp*
ultimately show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$)
∧ *strong-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$)
∧ *strong-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$)
and *A3*: *strong-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
by *blast*
from *A2* **obtain** *TRel* **where** ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$
and ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^*$
using *target-relation-from-source-target-relation*[**where** *Rel*=*Rel*]
by *blast*
with *A1 A3*
have *strongly-operational-sound* ($TRel^*$) ∧ *strong-reduction-simulation* ($(TRel^+)^{-1}$) *Target*
using *SOSou-iff-strong-reduction-simulation*[**where** *TRel*=*TRel*]
by *blast*
thus ∃ *TRel*. *strongly-operational-sound* ($TRel^*$) ∧ *strong-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* × (*'procS*, *'procT*) *Proc*) *set*
and *TRel* :: (*'procT* × *'procT*) *set*
assumes *enc*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *tRel*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^*$)
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* (⟦*S*⟧)) ∈ *Rel*
by *simp*
moreover assume ⟦*S*⟧ ⟼*Target*∗ *T*
hence *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*)∗ (*TargetTerm T*)
by (*simp add*: *STCal-steps*(*2*))
ultimately obtain *P′ Q′* **where** *A1*: *SourceTerm S* ⟼(*STCal Source Target*)∗ *P′*
and *A2*: *TargetTerm T* ⟼(*STCal Source Target*)∗ *Q′* **and** *A3*: (*P′*, *Q′*) ∈ *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 × 'procT) set
 shows (weakly-operational-corresponding (TRel*)
       ∧ 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 ∀ P Q Q′. P ≲⟦·⟧RT<TRel> Q ∧ Q ⟼(STCal Source Target)∗ Q′
      ⟶ (∃ P″ Q″. P ⟼(STCal Source Target)∗ P″ ∧ Q′ ⟼(STCal Source Target)∗ Q″
      ∧ P″ ≲⟦·⟧RT<TRel> Q″)
  proof clarify
   fix P Q Q′
   assume P ≲⟦·⟧RT<TRel> Q and Q ⟼(STCal Source Target)∗ Q′
   thus ∃ P″ Q″. P ⟼(STCal Source Target)∗ P″ ∧ Q′ ⟼(STCal Source Target)∗ Q″
       ∧ P″ ≲⟦·⟧RT<TRel> Q″
   proof (induct arbitrary: Q′)
    case (encR S)
    assume TargetTerm (⟦S⟧) ⟼(STCal Source Target)∗ Q′
    from this obtain T where A1: T ∈T Q′ and A2: ⟦S⟧ ⟼Target∗ T
     by (auto simp add: STCal-steps(2))
    from A2 woc obtain S′ T′ where A3: S ⟼Source∗ S′ and A4: T ⟼Target∗ T′
                            and A5: (⟦S′⟧, T′) ∈ TRel*
     by blast
    from A3 have SourceTerm S ⟼(STCal Source Target)∗ (SourceTerm S′)
     by (simp add: STCal-steps(1))
    moreover from A4 have TargetTerm T ⟼(STCal Source Target)∗ (TargetTerm T′)
     by (simp add: STCal-steps(2))
    moreover have SourceTerm S′ ≲⟦·⟧RT<TRel> TargetTerm T′
    proof −
     have A6: SourceTerm S′ ≲⟦·⟧RT<TRel> TargetTerm (⟦S′⟧)
      by (rule indRelRTPO.encR)
     from A5 have ⟦S′⟧ = T′ ∨ (⟦S′⟧, T′) ∈ TRel⁺
        using rtrancl-eq-or-trancl[of ⟦S′⟧ T′ TRel]
      by blast
     moreover from A6 have ⟦S′⟧ = T′ ⟹ SourceTerm S′ ≲⟦·⟧RT<TRel> TargetTerm T′
      by simp
     moreover have (⟦S′⟧, T′) ∈ TRel⁺ ⟹ SourceTerm S′ ≲⟦·⟧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 $(simp\ add\colon transitive\text{-}closure\text{-}of\text{-}TRel\text{-}to\text{-}indRelRTPO$[**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 $\exists P''\ Q''.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)* \ P''$
$\wedge\ Q' \longmapsto(STCal\ Source\ Target)*\ Q'' \wedge P'' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q''$
using $A1$
by *blast*
next
case $(source\ S)$
assume $B1\colon SourceTerm\ S \longmapsto(STCal\ Source\ Target)*\ Q'$
moreover have $Q' \longmapsto(STCal\ Source\ Target)*\ Q'$
by $(rule\ steps\text{-}refl)$
moreover from $B1$ **obtain** S' **where** $S' \in S\ Q'$
by $(auto\ simp\ add\colon STCal\text{-}steps(1))$
hence $Q' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q'$
by $(simp\ add\colon indRelRTPO.source)$
ultimately show $\exists P''\ Q''.\ SourceTerm\ S \longmapsto(STCal\ Source\ Target)*\ P''$
$\wedge\ Q' \longmapsto(STCal\ Source\ Target)*\ Q'' \wedge P'' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q''$
by *blast*
next
case $(target\ T1\ T2)$
assume $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)*\ Q'$
from *this* **obtain** $T2'$ **where** $C1\colon T2' \in T\ Q'$ **and** $C2\colon T2 \longmapsto Target*\ T2'$
by $(auto\ simp\ add\colon STCal\text{-}steps(2))$
assume $(T1,\ T2) \in TRel$
hence $(T1,\ T2) \in TRel^+$
by *simp*
with $C2\ csi$ **obtain** $T1'\ T2''$ **where** $C3\colon T1 \longmapsto Target*\ T1'$ **and** $C4\colon T2' \longmapsto Target*\ T2''$
and $C5\colon (T1',\ T2'') \in TRel^+$
by *blast*
from $C3$ **have** $TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)*\ (TargetTerm\ T1')$
by $(simp\ add\colon STCal\text{-}steps(2))$
moreover from $C1\ C4$ **have** $Q' \longmapsto(STCal\ Source\ Target)*\ (TargetTerm\ T2'')$
by $(simp\ add\colon STCal\text{-}steps(2))$
moreover from $C5$ **have** $TargetTerm\ T1' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ (TargetTerm\ T2'')$
by $(simp\ add\colon transitive\text{-}closure\text{-}of\text{-}TRel\text{-}to\text{-}indRelRTPO)$
ultimately show $\exists P''\ Q''.\ TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)*\ P''$
$\wedge\ Q' \longmapsto(STCal\ Source\ Target)*\ Q'' \wedge P'' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q''$
by *blast*
next
case $(trans\ P\ Q\ R\ R')$
assume $R \longmapsto(STCal\ Source\ Target)*\ R'$
and $\bigwedge R'.\ R \longmapsto(STCal\ Source\ Target)*\ R' \Longrightarrow \exists Q''\ R''.\ Q \longmapsto(STCal\ Source\ Target)*\ Q''$
$\wedge\ R' \longmapsto(STCal\ Source\ Target)*\ R'' \wedge Q'' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ R''$
and $\bigwedge Q'.\ Q \longmapsto(STCal\ Source\ Target)*\ Q' \Longrightarrow \exists P''\ Q''.\ P \longmapsto(STCal\ Source\ Target)*\ P''$
$\wedge\ Q' \longmapsto(STCal\ Source\ Target)*\ Q'' \wedge P'' \lesssim\llbracket\cdot\rrbracket RT{<}TRel{>}\ Q''$
moreover have $trans\ (indRelRTPO\ TRel)$
using $indRelRTPO.trans$
unfolding $trans\text{-}def$
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*^*)
∧ *weak-reduction-correspondence-simulation* (*TRel*^+) *Target*
proof
have ∀ *S*. *SourceTerm S* ≲⟦·⟧*RT*<*TRel*> *TargetTerm* (⟦*S*⟧)
by (*simp add*: *indRelRTPO.encR*)
moreover have ∀ *S T*. *SourceTerm S* ≲⟦·⟧*RT*<*TRel*> *TargetTerm T* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*^*
using *indRelRTPO-to-TRel*(*2*)[**where** *TRel*=*TRel*] *trans-closure-of-TRel-refl-cond*
by *simp*
ultimately show *weakly-operational-corresponding* (*TRel*^*)
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*^+) *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* × *'procT*) *set*
shows (*weakly-operational-corresponding* (*TRel*^*)
∧ *weak-reduction-correspondence-simulation* (*TRel*^+) *Target*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*^+)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*^*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*))
proof (*rule iffI*, *erule conjE*)
have ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelRTPO TRel*
by (*simp add*: *indRelRTPO.encR*)
moreover have ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ *TargetTerm T1* ≲⟦·⟧*RT*<*TRel*> *TargetTerm T2*
by (*simp add*: *indRelRTPO.target*)
moreover have ∀ *T1 T2*. *TargetTerm T1* ≲⟦·⟧*RT*<*TRel*> *TargetTerm T2* ⟶ (*T1*, *T2*) ∈ *TRel*^+
using *indRelRTPO-to-TRel*(*4*)[**where** *TRel*=*TRel*]
by *simp*
moreover have ∀ *S T*. *SourceTerm S* ≲⟦·⟧*RT*<*TRel*> *TargetTerm T* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*^*
using *indRelRTPO-to-TRel*(*2*)[**where** *TRel*=*TRel*] *trans-closure-of-TRel-refl-cond*
by *simp*
moreover assume *weakly-operational-corresponding* (*TRel*^*)
and *weak-reduction-correspondence-simulation* (*TRel*^+) *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*^+)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*^*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*)
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ *TRel*^+)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*^*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*

```
    and A2: ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
    and A3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
    and A4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*
    and A5: 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*)
                ∧ weak-reduction-correspondence-simulation (TRel⁺) Target
    by simp
qed

lemma rel-includes-TRel-modulo-preorder:
  fixes Rel  :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
    and TRel :: ('procT × 'procT) set
  assumes transT: trans TRel
  shows ((∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
          ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺))
         = (TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel})
proof (rule iffI, erule conjE)
  assume ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
     and ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
  with transT show TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
      using trancl-id[of TRel]
    by blast
next
  assume A: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
  hence ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
    by simp
  moreover from transT A
  have ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
      using trancl-id[of TRel]
    by blast
  ultimately show (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
                ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
    by simp
qed

lemma (in encoding) WOC-wrt-preorder-iff-reduction-correspondence-simulation:
  fixes TRel :: ('procT × 'procT) set
  shows (weakly-operational-corresponding TRel ∧ preorder TRel
         ∧ weak-reduction-correspondence-simulation TRel Target)
         = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
            ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
            ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
            ∧ preorder Rel
            ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target))
proof (rule iffI, erule conjE, erule conjE, erule conjE)
  assume A1: operational-complete TRel and A2: weakly-operational-sound TRel
     and A3:preorder TRel and A4: 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 ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelRTPO TRel
    by (simp add: indRelRTPO.encR)
  moreover
  have TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ indRelRTPO TRel}
  proof auto
    fix TP TQ
    assume (TP, TQ) ∈ TRel
    thus TargetTerm TP ≲⟦·⟧RT<TRel> TargetTerm TQ
      by (rule indRelRTPO.target)
  next
    fix TP TQ
    assume TargetTerm TP ≲⟦·⟧RT<TRel> TargetTerm TQ
    with A3 show (TP, TQ) ∈ TRel
        using indRelRTPO-to-TRel(4)[where TRel=TRel] trancl-id[of TRel]
        unfolding preorder-on-def
      by blast
  qed
  moreover from A3
  have ∀ S T. (SourceTerm S, TargetTerm T) ∈ indRelRTPO TRel ⟶ (⟦S⟧, T) ∈ 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 ∀ S T. (SourceTerm S, TargetTerm T) ∈ indRelRTPO TRel ⟶ (⟦S⟧, T) ∈ 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                  ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
                  ∧ preorder Rel
                  ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)
      unfolding preorder-on-def
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
         ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
         ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
         ∧ preorder Rel
         ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)
  from this obtain Rel where B1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
   and B2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
   and B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ 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* ($TRel^+$) *Target*)
= *weak-reduction-bisimulation* (*indRelRTPO TRel*) (*STCal Source Target*)
proof (*rule iffI*, *erule conjE*)
assume *ocorr*: *operational-corresponding* ($TRel^*$)
and *bisim*: *weak-reduction-bisimulation* ($TRel^+$) *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* ($(TRel^+)^{-1}$) *Target*
using *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where** $Rel=TRel^+$]
by *simp*
with *ocorr* **have** *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 *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* ($TRel^+$) *Target*
using *OCom-iff-indRelRTPO-is-weak-reduction-simulation*[**where** *TRel=TRel*]

by *simp*
moreover from *bisim*
have *weak-reduction-simulation* $((\mathit{indRelRTPO\ TRel})^{-1})$ $(\mathit{STCal\ Source\ Target})$
using *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where** *Rel=indRelRTPO TRel*]
by *simp*
hence *operational-sound* $(\mathit{TRel}^*) \wedge$ *weak-reduction-simulation* $((\mathit{TRel}^+)^{-1})$ *Target*
using *OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation*[**where** *TRel=TRel*]
by *simp*
ultimately show *operational-corresponding* $(\mathit{TRel}^*) \wedge$ *weak-reduction-bisimulation* (TRel^+) *Target*
using *weak-reduction-simulations-impl-bisimulation*[**where** $\mathit{Rel}=\mathit{TRel}^+$]
by *simp*
qed

lemma (**in** *encoding*) *OC-iff-weak-reduction-bisimulation*:
fixes *TRel* :: $('\mathit{procT} \times '\mathit{procT})$ *set*
shows $(\mathit{operational\text{-}corresponding}\ (\mathit{TRel}^*) \wedge \mathit{weak\text{-}reduction\text{-}bisimulation}\ (\mathit{TRel}^+)\ \mathit{Target})$
$= (\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
$\wedge (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*))
proof (*rule iffI*, *erule conjE*)
have $\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{indRelRTPO\ TRel}$
by (*simp add*: *indRelRTPO.encR*)
moreover have $\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT\text{<}\mathit{TRel}\text{>}\ \mathit{TargetTerm}\ T2$
by (*simp add*: *indRelRTPO.target*)
moreover have $\forall T1\ T2.\ \mathit{TargetTerm}\ T1 \lesssim[\![\cdot]\!]RT\text{<}\mathit{TRel}\text{>}\ \mathit{TargetTerm}\ T2 \longrightarrow (T1,\ T2) \in \mathit{TRel}^+$
using *indRelRTPO-to-TRel*(*4*)[**where** *TRel=TRel*]
by *simp*
moreover have $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT\text{<}\mathit{TRel}\text{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$
using *indRelRTPO-to-TRel*(*2*)[**where** *TRel=TRel*] *trans-closure-of-TRel-refl-cond*
by *simp*
moreover assume *operational-corresponding* (TRel^*)
and *weak-reduction-bisimulation* (TRel^+) *Target*
hence *weak-reduction-bisimulation* (*indRelRTPO TRel*) (*STCal Source Target*)
using *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*[**where** *TRel=TRel*]
by *simp*
ultimately show $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
$\wedge (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
next
assume $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
$\wedge (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: $\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel}$
and *A2*: $\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}$
and *A3*: $\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+$
and *A4*: $\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$
and *A5*: *weak-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *operational-complete* (TRel^*)
$\wedge$ *weak-reduction-simulation* (TRel^+) *Target*
using *OCom-iff-weak-reduction-simulation*[**where** *TRel=TRel*]
by *blast*
moreover from *A5* **have** *weak-reduction-simulation* (Rel^{-1}) (*STCal Source Target*)
using *weak-reduction-bisimulations-impl-inverse-is-simulation*[**where** *Rel=Rel*]

by *simp*
with *A1 A2 A3 A4* **have** *operational-sound* ($TRel^*$)
∧ *weak-reduction-simulation* (($TRel^+$)$^{-1}$) *Target*
using *OSou-iff-weak-reduction-simulation*[**where** *TRel*=*TRel*]
by *blast*
ultimately show *operational-corresponding* ($TRel^*$)
∧ *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* × '*procT*) *set*
shows (*operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-bisimulation TRel Target*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *preorder Rel*
∧ *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 ∀ *S*. *SourceTerm S* $\lesssim$⟦·⟧*RT*<*TRel*> *TargetTerm* (⟦*S*⟧)
by (*simp add*: *indRelRTPO.encR*)
moreover
have *TRel* = {(*T1*, *T2*). *TargetTerm T1* $\lesssim$⟦·⟧*RT*<*TRel*> *TargetTerm T2*}
proof *auto*
fix *TP TQ*
assume (*TP*, *TQ*) ∈ *TRel*
thus *TargetTerm TP* $\lesssim$⟦·⟧*RT*<*TRel*> *TargetTerm TQ*
by (*rule indRelRTPO.target*)
next
fix *TP TQ*
assume *TargetTerm TP* $\lesssim$⟦·⟧*RT*<*TRel*> *TargetTerm TQ*
with *A3* **show** (*TP*, *TQ*) ∈ *TRel*
using *indRelRTPO-to-TRel*(*4*)[**where** *TRel*=*TRel*] *trancl-id*[*of TRel*]
unfolding *preorder-on-def*
by *blast*
qed
moreover from *A3*
have ∀ *S T*. *SourceTerm S* $\lesssim$⟦·⟧*RT*<*TRel*> *TargetTerm T* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^+$
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** $\forall S\ T.\ \mathit{SourceTerm}\ S \lesssim[\![\cdot]\!]RT\text{<}TRel\text{>}\ \mathit{TargetTerm}\ T \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}$
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 $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge\ \mathit{TRel} = \{(T1,\ T2).\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}\}$
$\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel})$
$\wedge$ *preorder Rel*
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
unfolding *preorder-on-def*
by *blast*
next
assume $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge\ \mathit{TRel} = \{(T1,\ T2).\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}\}$
$\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel})$
$\wedge$ *preorder Rel*
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *B1*: $\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel}$
and *B2*: $\mathit{TRel} = \{(T1,\ T2).\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}\}$
and *B3*: $\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}$ **and** *B4*: *preorder Rel*
and *B5*: *weak-reduction-bisimulation 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*: $\mathit{TRel}^+ = \mathit{TRel}$
using *trancl-id*[*of TRel*]
by *simp*
with *B6* **have** *B9*: $\mathit{TRel}^* = \mathit{TRel}$
using *reflcl-trancl*[*of TRel*]
unfolding *refl-on-def*
by *blast*
with *B3* **have** $\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*$
by *simp*
moreover from *B2 B8* **have** $\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel}$
and $\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+$
by *auto*
ultimately have $\exists \mathit{Rel}.\ (\forall S.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ ([\![S]\!])) \in \mathit{Rel})$
$\wedge\ (\forall T1\ T2.\ (T1,\ T2) \in \mathit{TRel} \longrightarrow (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel})$
$\wedge\ (\forall T1\ T2.\ (\mathit{TargetTerm}\ T1,\ \mathit{TargetTerm}\ T2) \in \mathit{Rel} \longrightarrow (T1,\ T2) \in \mathit{TRel}^+)$
$\wedge\ (\forall S\ T.\ (\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ T) \in \mathit{Rel} \longrightarrow ([\![S]\!],\ T) \in \mathit{TRel}^*)$
$\wedge$ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
using *B1 B5*
by *blast*
hence *operational-corresponding* (TRel^*)
$\wedge$ *weak-reduction-bisimulation* (TRel^+) *Target*
using *OC-iff-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
with *B8 B9* **have** *operational-corresponding TRel* $\wedge$ *weak-reduction-bisimulation TRel Target*
by *simp*

moreover from *B6 B7* **have** *preorder TRel*
unfolding *preorder-on-def*
by *blast*
ultimately show *operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-bisimulation TRel Target*
by *blast*
qed

lemma (**in** *encoding*) *OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation*:
fixes *TRel* :: (*'procT* × *'procT*) *set*
assumes *eqT*: *equivalence TRel*
shows (*operational-corresponding TRel* ∧ *weak-reduction-bisimulation TRel Target*) ⟷
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* ∼⟦·⟧*T*<*TRel*> *Q* **and** *P* ⟼(*STCal Source Target*)∗ *P'*
thus ∃ *Q'*. *Q* ⟼(*STCal Source Target*)∗ *Q'* ∧ *P'* ∼⟦·⟧*T*<*TRel*> *Q'*
proof (*induct arbitrary*: *P'*)
case (*encR S*)
assume *SourceTerm S* ⟼(*STCal Source Target*)∗ *P'*
from *this* **obtain** *S'* **where** *A1*: *S* ⟼*Source*∗ *S'* **and** *A2*: *S'* ∈*S P'*
by (*auto simp add*: *STCal-steps*(*1*))
from *A1 oc* **obtain** *T* **where** *A3*: ⟦*S*⟧ ⟼*Target*∗ *T* **and** *A4*: (⟦*S'*⟧, *T*) ∈ *TRel*
by *blast*
from *A3* **have** *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*)∗ (*TargetTerm T*)
by (*simp add*: *STCal-steps*(*2*))
moreover have *P'* ∼⟦·⟧*T*<*TRel*> *TargetTerm T*
proof −
from *A2* **have** *P'* ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S'*⟧)
by (*simp add*: *indRelTEQ.encR*)
moreover from *A4* **have** *TargetTerm* (⟦*S'*⟧) ∼⟦·⟧*T*<*TRel*> *TargetTerm T*
by (*rule indRelTEQ.target*)
ultimately show *P'* ∼⟦·⟧*T*<*TRel*> *TargetTerm T*
by (*rule indRelTEQ.trans*)
qed
ultimately show ∃ *Q'*. *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*)∗ *Q'* ∧ *P'* ∼⟦·⟧*T*<*TRel*> *Q'*
by *blast*
next
case (*encL S*)
assume *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*)∗ *P'*
from *this* **obtain** *T* **where** *B1*: ⟦*S*⟧ ⟼*Target*∗ *T* **and** *B2*: *T* ∈*T P'*
by (*auto simp add*: *STCal-steps*(*2*))
from *B1 oc* **obtain** *S'* **where** *B3*: *S* ⟼*Source*∗ *S'* **and** *B4*: (⟦*S'*⟧, *T*) ∈ *TRel*
by *blast*
from *B3* **have** *SourceTerm S* ⟼(*STCal Source Target*)∗ (*SourceTerm S'*)
by (*simp add*: *STCal-steps*(*1*))
moreover have *P'* ∼⟦·⟧*T*<*TRel*> *SourceTerm S'*
proof −
from *B4 eqT* **have** (*T*, ⟦*S'*⟧) ∈ *TRel*
unfolding *equiv-def sym-def*
by *blast*
with *B2* **have** *P'* ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S'*⟧)
by (*simp add*: *indRelTEQ.target*)
moreover have *TargetTerm* (⟦*S'*⟧) ∼⟦·⟧*T*<*TRel*> *SourceTerm S'*
by (*rule indRelTEQ.encL*)
ultimately show *P'* ∼⟦·⟧*T*<*TRel*> *SourceTerm S'*
by (*rule indRelTEQ.trans*)
qed

ultimately show $\exists\, Q'.\ \mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})*\ Q' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by *blast*
next
case (*target T1 T2*)
assume $\mathit{TargetTerm}\ T1 \longmapsto(\mathit{STCal\ Source\ Target})*\ P'$
from *this* **obtain** $T1'$ **where** $C1$: $T1 \longmapsto \mathit{Target}*\ T1'$ **and** $C2$: $T1' \in T\ P'$
by (*auto simp add*: *STCal-steps*(*2*))
assume $(T1,\ T2) \in \mathit{TRel}$
with *C1 bisimT* **obtain** $T2'$ **where** $C3$: $T2 \longmapsto \mathit{Target}*\ T2'$ **and** $C4$: $(T1',\ T2') \in \mathit{TRel}$
by *blast*
from *C3* **have** $\mathit{TargetTerm}\ T2 \longmapsto(\mathit{STCal\ Source\ Target})*\ (\mathit{TargetTerm}\ T2')$
by (*simp add*: *STCal-steps*(*2*))
moreover from *C2 C4* **have** $P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T2'$
by (*simp add*: *indRelTEQ.target*)
ultimately show $\exists\, Q'.\ \mathit{TargetTerm}\ T2 \longmapsto(\mathit{STCal\ Source\ Target})*\ Q' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by *blast*
next
case (*trans P Q R*)
assume $P \longmapsto(\mathit{STCal\ Source\ Target})*\ P'$
and $\bigwedge P'.\ P \longmapsto(\mathit{STCal\ Source\ Target})*\ P'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto(\mathit{STCal\ Source\ Target})*\ Q' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
from *this* **obtain** Q' **where** $D1$: $Q \longmapsto(\mathit{STCal\ Source\ Target})*\ Q'$ **and** $D2$: $P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(\mathit{STCal\ Source\ Target})*\ Q'$
$\Longrightarrow \exists\, R'.\ R \longmapsto(\mathit{STCal\ Source\ Target})*\ R' \wedge Q' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ R'$
with *D1* **obtain** R' **where** $D3$: $R \longmapsto(\mathit{STCal\ Source\ Target})*\ R'$ **and** $D4$: $Q' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ R'$
by *blast*
from *D2 D4* **have** $P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ R'$
by (*rule indRelTEQ.trans*)
with *D3* **show** $\exists\, R'.\ R \longmapsto(\mathit{STCal\ Source\ Target})*\ R' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ R'$
by *blast*
qed
next
fix $P\ Q\ Q'$
assume $P \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q$ **and** $Q \longmapsto(\mathit{STCal\ Source\ Target})*\ Q'$
thus $\exists\, P'.\ P \longmapsto(\mathit{STCal\ Source\ Target})*\ P' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
proof (*induct arbitrary*: Q')
case (*encR S*)
assume $\mathit{TargetTerm}\ (\llbracket S\rrbracket) \longmapsto(\mathit{STCal\ Source\ Target})*\ Q'$
from *this* **obtain** T **where** $E1$: $\llbracket S\rrbracket \longmapsto \mathit{Target}*\ T$ **and** $E2$: $T \in T\ Q'$
by (*auto simp add*: *STCal-steps*(*2*))
from *E1 oc* **obtain** S' **where** $E3$: $S \longmapsto \mathit{Source}*\ S'$ **and** $E4$: $(\llbracket S'\rrbracket,\ T) \in \mathit{TRel}$
by *blast*
from *E3* **have** $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})*\ (\mathit{SourceTerm}\ S')$
by (*simp add*: *STCal-steps*(*1*))
moreover have $\mathit{SourceTerm}\ S' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
proof −
have $\mathit{SourceTerm}\ S' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ (\llbracket S'\rrbracket)$
by (*rule indRelTEQ.encR*)
moreover from *E2 E4* **have** $\mathit{TargetTerm}\ (\llbracket S'\rrbracket) \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by (*simp add*: *indRelTEQ.target*)
ultimately show $\mathit{SourceTerm}\ S' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by (*rule indRelTEQ.trans*)
qed
ultimately show $\exists\, P'.\ \mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})*\ P' \wedge P' \sim\llbracket\cdot\rrbracket T{<}\mathit{TRel}{>}\ Q'$
by *blast*
next
case (*encL S*)
assume $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})*\ Q'$
from *this* **obtain** S' **where** $F1$: $S \longmapsto \mathit{Source}*\ S'$ **and** $F2$: $S' \in S\ Q'$
by (*auto simp add*: *STCal-steps*(*1*))

from *F1 oc* **obtain** *T* **where** *F3*: $[\![S]\!] \longmapsto Target*\ T$ **and** *F4*: $([\![S']\!],\ T) \in TRel$
by *blast*
from *F3* **have** *TargetTerm* $([\![S]\!]) \longmapsto (STCal\ Source\ Target)*\ (TargetTerm\ T)$
by (*simp add*: *STCal-steps*(*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 $\exists P'.\ TargetTerm\ ([\![S]\!]) \longmapsto (STCal\ Source\ Target)*\ P' \land P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
next
case (*target T1 T2*)
assume $TargetTerm\ T2 \longmapsto (STCal\ Source\ Target)*\ Q'$
from *this* **obtain** $T2'$ **where** *G1*: $T2 \longmapsto Target*\ T2'$ **and** *G2*: $T2' \in T\ Q'$
by (*auto simp add*: *STCal-steps*(*2*))
assume $(T1,\ T2) \in TRel$
with *G1 bisimT* **obtain** $T1'$ **where** *G3*: $T1 \longmapsto Target*\ T1'$ **and** *G4*: $(T1',\ T2') \in TRel$
by *blast*
from *G3* **have** $TargetTerm\ T1 \longmapsto (STCal\ Source\ Target)*\ (TargetTerm\ T1')$
by (*simp add*: *STCal-steps*(*2*))
moreover from *G2 G4* **have** $TargetTerm\ T1' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by (*simp add*: *indRelTEQ.target*)
ultimately show $\exists P'.\ TargetTerm\ T1 \longmapsto (STCal\ Source\ Target)*\ P' \land P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
next
case (*trans P Q R R'*)
assume $R \longmapsto (STCal\ Source\ Target)*\ R'$
and $\bigwedge R'.\ R \longmapsto (STCal\ Source\ Target)*\ R'$
$\Longrightarrow \exists Q'.\ Q \longmapsto (STCal\ Source\ Target)*\ Q' \land Q' \sim[\![\cdot]\!]T{<}TRel{>}\ R'$
from *this* **obtain** Q' **where** *H1*: $Q \longmapsto (STCal\ Source\ Target)*\ Q'$ **and** *H2*: $Q' \sim[\![\cdot]\!]T{<}TRel{>}\ R'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto (STCal\ Source\ Target)*\ Q'$
$\Longrightarrow \exists P'.\ P \longmapsto (STCal\ Source\ Target)*\ P' \land P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
with *H1* **obtain** P' **where** *H3*: $P \longmapsto (STCal\ Source\ Target)*\ P'$ **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** $\exists P'.\ P \longmapsto (STCal\ Source\ Target)*\ P' \land P' \sim[\![\cdot]\!]T{<}TRel{>}\ R'$
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 $SourceTerm\ S \sim[\![\cdot]\!]T{<}TRel{>}$ *TargetTerm* $([\![S]\!])$
by (*rule indRelTEQ.encR*)
moreover assume $S \longmapsto Source*\ S'$
hence $SourceTerm\ S \longmapsto (STCal\ Source\ Target)*\ (SourceTerm\ S')$
by (*simp add*: *STCal-steps*(*1*))
ultimately obtain Q' **where** *I1*: $TargetTerm\ ([\![S]\!]) \longmapsto (STCal\ Source\ Target)*\ Q'$
and *I2*: $SourceTerm\ S' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$

using *bisim*
by *blast*
from *I1* **obtain** T **where** *I3*: $[\![S]\!] \longmapsto Target{*}\ T$ **and** *I4*: $T \in T\ Q'$
by (*auto simp add*: *STCal-steps*(*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** $\exists\, T.\ [\![S]\!] \longmapsto Target{*}\ T \wedge ([\![S']\!],\ T) \in TRel$
by *blast*
next
fix *S T*
have *SourceTerm S* $\sim[\![\cdot]\!]T$<*TRel*> *TargetTerm* ($[\![S]\!]$)
by (*rule indRelTEQ.encR*)
moreover assume $[\![S]\!] \longmapsto Target{*}\ T$
hence *TargetTerm* ($[\![S]\!]$) $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm T*)
by (*simp add*: *STCal-steps*(*2*))
ultimately obtain Q' **where** *J1*: *SourceTerm S* $\longmapsto$(*STCal Source Target*)$*$ Q'
and *J2*: $Q' \sim[\![\cdot]\!]T$<*TRel*> *TargetTerm T*
using *bisim*
by *blast*
from *J1* **obtain** S' **where** *J3*: $S \longmapsto Source{*}\ S'$ **and** *J4*: $S' \in S\ Q'$
by (*auto simp add*: *STCal-steps*(*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 *J3* **show** $\exists\, S'.\ S \longmapsto Source{*}\ S' \wedge ([\![S']\!],\ T) \in TRel$
by *blast*
qed
moreover have *weak-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** *weak-reduction-bisimulation TRel Target*
using *indRelTEQ-impl-TRel-is-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
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*) $\longleftrightarrow$ ($\exists$ *Rel*.
($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel* $\wedge$ (*TargetTerm* ($[\![S]\!]$), *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*))
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* ($[\![S]\!]$) $\wedge$ *TargetTerm* ($[\![S]\!]$) $\sim[\![\cdot]\!]T<TRel>$ *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* ($[\![S]\!]$)) $\in$ *Rel* $\wedge$ (*TargetTerm* ($[\![S]\!]$), *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* ($[\![S]\!]$)) $\in$ *Rel*
   $\wedge$ (*TargetTerm* ($[\![S]\!]$), *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* ($[\![S]\!]$)) $\in$ *Rel*
   $\wedge$ (*TargetTerm* ($[\![S]\!]$), *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* ($[\![S]\!]$)) $\in$ *Rel*
  by *simp*
 moreover assume $S \longmapsto Source* S'$
 hence *SourceTerm S* $\longmapsto$(*STCal Source Target*)$*$ (*SourceTerm S'*)
  by (*simp add*: *STCal-steps*(*1*))
 ultimately obtain *Q'* **where** *B1*: *TargetTerm* ($[\![S]\!]$) $\longmapsto$(*STCal Source Target*)$*$ *Q'*
   and *B2*: (*SourceTerm S'*, *Q'*) $\in$ *Rel*
   using *A4*
  by *blast*
 from *B1* **obtain** *T* **where** *B3*: $[\![S]\!] \longmapsto Target* T$ **and** *B4*: $T \in T\ Q'$
  by (*auto simp add*: *STCal-steps*(*2*))
 from *A1* **have** (*TargetTerm* ($[\![S']\!]$), *SourceTerm S'*) $\in$ *Rel*
  by *simp*
 with *B2 A3* **have** (*TargetTerm* ($[\![S']\!]$), *Q'*) $\in$ *Rel*
   unfolding *trans-def*
  by *blast*
 with *B4 A2* **have** ($[\![S']\!]$, *T*) $\in$ *TRel*
  by *simp*
 with *B3* **show** $\exists T$. $[\![S]\!] \longmapsto Target* T$ $\wedge$ ($[\![S']\!]$, *T*) $\in$ *TRel*
  by *blast*
next
 fix *S T*
 from *A1* **have** (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*
  by *simp*
 moreover assume $[\![S]\!] \longmapsto Target* T$

hence *TargetTerm* ($[\![S]\!]$) $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm T*)
  by (*simp add*: *STCal-steps*(*2*))
ultimately obtain P' **where** *C1*: *SourceTerm S* $\longmapsto$(*STCal Source Target*)$*$ P'
  and *C2*: (P', *TargetTerm T*) $\in$ *Rel*
    using *A4*
  by *blast*
from *C1* **obtain** S' **where** *C3*: $S \longmapsto$*Source*$*$ S' **and** *C4*: $S' \in S\ P'$
  by (*auto simp add*: *STCal-steps*(*1*))
from *A1 C4* **have** (*TargetTerm* ($[\![S']\!]$), P') $\in$ *Rel*
  by *simp*
from *A3 this C2* **have** (*TargetTerm* ($[\![S']\!]$), *TargetTerm T*) $\in$ *Rel*
    unfolding *trans-def*
  by *blast*
with *A2* **have** ($[\![S']\!]$, *T*) $\in$ *TRel*
  by *simp*
with *C3* **show** $\exists S'$. $S \longmapsto$*Source*$*$ S' $\wedge$ ($[\![S']\!]$, *T*) $\in$ *TRel*
  by *blast*
qed
moreover have *weak-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* $\longmapsto$*Target*$*$ *TP'*
hence *TargetTerm TP* $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm TP'*)
  by (*simp add*: *STCal-steps*(*2*))
ultimately obtain Q' **where** *D1*: *TargetTerm TQ* $\longmapsto$(*STCal Source Target*)$*$ Q'
  and *D2*: (*TargetTerm TP'*, Q') $\in$ *Rel*
    using *A4*
  by *blast*
from *D1* **obtain** *TQ'* **where** *D3*: *TQ* $\longmapsto$*Target*$*$ *TQ'* **and** *D4*: *TQ'* $\in T$ Q'
  by (*auto simp add*: *STCal-steps*(*2*))
from *A2 D2 D4* **have** (*TP'*, *TQ'*) $\in$ *TRel*
  by *simp*
with *D3* **show** $\exists$ *TQ'*. *TQ* $\longmapsto$*Target*$*$ *TQ'* $\wedge$ (*TP'*, *TQ'*) $\in$ *TRel*
  by *blast*
next
fix *TP TQ TQ'*
assume (*TP*, *TQ*) $\in$ *TRel*
with *A2* **have** (*TargetTerm TP*, *TargetTerm TQ*) $\in$ *Rel*
  by *simp*
moreover assume *TQ* $\longmapsto$*Target*$*$ *TQ'*
hence *TargetTerm TQ* $\longmapsto$(*STCal Source Target*)$*$ (*TargetTerm TQ'*)
  by (*simp add*: *STCal-steps*(*2*))
ultimately obtain P' **where** *E1*: *TargetTerm TP* $\longmapsto$(*STCal Source Target*)$*$ P'
  and *E2*: (P', *TargetTerm TQ'*) $\in$ *Rel*
    using *A4*
  by *blast*
from *E1* **obtain** *TP'* **where** *E3*: *TP* $\longmapsto$*Target*$*$ *TP'* **and** *E4*: *TP'* $\in T$ P'
  by (*auto simp add*: *STCal-steps*(*2*))
from *A2 E2 E4* **have** (*TP'*, *TQ'*) $\in$ *TRel*
  by *simp*
with *E3* **show** $\exists$ *TP'*. *TP* $\longmapsto$*Target*$*$ *TP'* $\wedge$ (*TP'*, *TQ'*) $\in$ *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 *SOCom-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 *SOCom-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*)
$= (\exists\, Rel.\ (\forall\, S.\ (SourceTerm\ S,\ TargetTerm\ ([\![S]\!])) \in Rel)$
$\wedge\ (\forall\, T1\ T2.\ (T1,\ T2) \in TRel \longrightarrow (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel)$
$\wedge\ (\forall\, T1\ T2.\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel \longrightarrow (T1,\ T2) \in TRel^+)$
$\wedge\ (\forall\, S\ T.\ (SourceTerm\ S,\ TargetTerm\ T) \in Rel \longrightarrow ([\![S]\!],\ T) \in TRel^*)$
$\wedge$ *strong-reduction-bisimulation Rel* (*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 *strongly-operational-corresponding* ($TRel^*$)
and *strong-reduction-bisimulation* ($TRel^+$) *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*. (*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*) $\in$ $TRel^*$)
$\wedge$ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
next
assume $\exists$ *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*) $\in$ $TRel^*$)
$\wedge$ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: $\forall$ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) $\in$ *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* $\longrightarrow$ ([[*S*]], *T*) $\in$ $TRel^*$
and *A5*: *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *strongly-operational-complete* ($TRel^*$)
$\wedge$ *strong-reduction-simulation* ($TRel^+$) *Target*
using *SOCom-iff-strong-reduction-simulation*[**where** *TRel*=*TRel*]
by *blast*
moreover from *A5* **have** *strong-reduction-simulation* (Rel^{-1}) (*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* (($TRel^+$)$^{-1}$) *Target*
using *SOSou-iff-strong-reduction-simulation*[**where** *TRel*=*TRel*]
by *blast*
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-wrt-preorder-iff-strong-reduction-bisimulation*:
fixes *TRel* :: ('*procT* × '*procT*) *set*
shows (*strongly-operational-corresponding TRel* $\wedge$ *preorder TRel*
$\wedge$ *strong-reduction-bisimulation TRel Target*)
= ($\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) $\in$ *Rel*)
$\wedge$ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) $\in$ *Rel*}
$\wedge$ ($\forall$ *S T*. (*SourceTerm S*, *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*: *strongly-operational-complete TRel* **and** *A2*: *strongly-operational-sound TRel*
and *A3*:*preorder TRel* **and** *A4*: *strong-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 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 ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ indRelRTPO TRel
    by (simp add: indRelRTPO.encR)
  moreover
  have TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ indRelRTPO TRel}
  proof auto
    fix TP TQ
    assume (TP, TQ) ∈ TRel
    thus TargetTerm TP ≲⟦·⟧RT<TRel> TargetTerm TQ
      by (rule indRelRTPO.target)
  next
    fix TP TQ
    assume TargetTerm TP ≲⟦·⟧RT<TRel> TargetTerm TQ
    with A3 show (TP, TQ) ∈ TRel
        using indRelRTPO-to-TRel(4)[where TRel=TRel] trancl-id[of TRel]
        unfolding preorder-on-def
      by blast
  qed
  moreover from A3
  have ∀ S T. (SourceTerm S, TargetTerm T) ∈ indRelRTPO TRel ⟶ (⟦S⟧, T) ∈ 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 ∀ S T. (SourceTerm S, TargetTerm T) ∈ indRelRTPO TRel ⟶ (⟦S⟧, T) ∈ 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                  ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
                  ∧ preorder Rel
                  ∧ strong-reduction-bisimulation Rel (STCal Source Target)
      unfolding preorder-on-def
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
          ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
          ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
          ∧ preorder Rel
          ∧ strong-reduction-bisimulation Rel (STCal Source Target)
  from this obtain Rel where B1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
   and B2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
   and B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel and B4: preorder Rel
   and B5: strong-reduction-bisimulation 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*
by (*rule trancl-id*)
with *B6* **have** *B9*: $TRel^{*}$ = *TRel*
using *reflcl-trancl*[*of TRel*]
unfolding *refl-on-def*
by *blast*
with *B3* **have** ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^{*}$
by *simp*
moreover from *B2 B8* **have** ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^{+}$
by *auto*
ultimately have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^{+}$)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^{*}$)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
using *B1 B5*
by *blast*
hence *strongly-operational-corresponding* ($TRel^{*}$) ∧ *strong-reduction-bisimulation* ($TRel^{+}$) *Target*
using *SOC-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
with *B8 B9*
have *strongly-operational-corresponding TRel* ∧ *strong-reduction-bisimulation TRel Target*
by *simp*
moreover from *B6 B7* **have** *preorder TRel*
unfolding *preorder-on-def*
by *blast*
ultimately show *strongly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *strong-reduction-bisimulation TRel Target*
by *blast*
qed

lemma (**in** *encoding*) *SOC-wrt-TRel-iff-strong-reduction-bisimulation*:
shows (∃ *TRel*. *strongly-operational-corresponding* ($TRel^{*}$)
∧ *strong-reduction-bisimulation* ($TRel^{+}$) *Target*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^{=}$)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*))
proof (*rule iffI*)
assume ∃ *TRel*. *strongly-operational-corresponding* ($TRel^{*}$)
∧ *strong-reduction-bisimulation* ($TRel^{+}$) *Target*
from *this* **obtain** *TRel* **where** *strongly-operational-corresponding* ($TRel^{*}$)
and *strong-reduction-bisimulation* ($TRel^{+}$) *Target*
by *blast*
hence *strong-reduction-bisimulation* (*indRelRTPO TRel*) (*STCal Source Target*)
using *SOC-iff-indRelRTPO-is-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *indRelRTPO TRel*
by (*simp add*: *indRelRTPO.encR*)
moreover have ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ (*indRelRTPO TRel*)
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $(indRelRTPO\ TRel)^{=}$
using *indRelRTPO-relates-source-target*[**where** *TRel*=*TRel*]
by *simp*
ultimately show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)

∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *A2*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel*
⟶ (*TargetTerm* (⟦*S*⟧), *TargetTerm T*) ∈ $Rel^=$
and *A3*: *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
from *A2* **obtain** *TRel* **where** ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
and ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$
and ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^*$
using *target-relation-from-source-target-relation*[**where** *Rel*=*Rel*]
by *blast*
with *A1 A3* **have** ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$)
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^*$)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *strongly-operational-corresponding* ($TRel^*$)
∧ *strong-reduction-bisimulation* ($TRel^+$) *Target*
using *SOC-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
thus ∃ *TRel*. *strongly-operational-corresponding* ($TRel^*$)
∧ *strong-reduction-bisimulation* ($TRel^+$) *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* ∧ *strong-reduction-bisimulation TRel Target*)
⟷ *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* ∼⟦·⟧*T*<*TRel*> *Q* **and** *P* ⟼(*STCal Source Target*) *P′*
thus ∃ *Q′*. *Q* ⟼(*STCal Source Target*) *Q′* ∧ *P′* ∼⟦·⟧*T*<*TRel*> *Q′*
proof (*induct arbitrary*: *P′*)
case (*encR S*)
assume *SourceTerm S* ⟼(*STCal Source Target*) *P′*
from *this* **obtain** *S′* **where** *A1*: *S* ⟼*Source S′* **and** *A2*: *S′* ∈*S P′*
by (*auto simp add*: *STCal-step*(*1*))
from *A1 oc* **obtain** *T* **where** *A3*: ⟦*S*⟧ ⟼*Target T* **and** *A4*: (⟦*S′*⟧, *T*) ∈ *TRel*
by *blast*
from *A3* **have** *TargetTerm* (⟦*S*⟧) ⟼(*STCal Source Target*) (*TargetTerm T*)
by (*simp add*: *STCal-step*(*2*))
moreover have *P′* ∼⟦·⟧*T*<*TRel*> *TargetTerm T*
proof −
from *A2* **have** *P′* ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S′*⟧)
by (*simp add*: *indRelTEQ.encR*)
moreover from *A4* **have** *TargetTerm* (⟦*S′*⟧) ∼⟦·⟧*T*<*TRel*> *TargetTerm T*
by (*rule indRelTEQ.target*)
ultimately show *P′* ∼⟦·⟧*T*<*TRel*> *TargetTerm T*

by (*rule indRelTEQ.trans*)
qed
ultimately show $\exists\, Q'.\ TargetTerm\ ([\![S]\!]) \longmapsto (STCal\ Source\ Target)\ Q' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
next
case (*encL S*)
assume $TargetTerm\ ([\![S]\!]) \longmapsto (STCal\ Source\ Target)\ P'$
from *this* **obtain** T **where** $B1$: $[\![S]\!] \longmapsto Target\ T$ **and** $B2$: $T \in T\ P'$
by (*auto simp add*: *STCal-step*(*2*))
from *B1 oc* **obtain** S' **where** $B3$: $S \longmapsto Source\ S'$ **and** $B4$: $([\![S']\!],\ T) \in TRel$
by *blast*
from *B3* **have** $SourceTerm\ S \longmapsto (STCal\ Source\ Target)\ (SourceTerm\ S')$
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 $\exists\, Q'.\ SourceTerm\ S \longmapsto (STCal\ Source\ Target)\ Q' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
next
case (*target T1 T2*)
assume $TargetTerm\ T1 \longmapsto (STCal\ Source\ Target)\ P'$
from *this* **obtain** $T1'$ **where** $C1$: $T1 \longmapsto Target\ T1'$ **and** $C2$: $T1' \in T\ P'$
by (*auto simp add*: *STCal-step*(*2*))
assume $(T1,\ T2) \in TRel$
with *C1 bisimT* **obtain** $T2'$ **where** $C3$: $T2 \longmapsto Target\ T2'$ **and** $C4$: $(T1',\ T2') \in TRel$
by *blast*
from *C3* **have** $TargetTerm\ T2 \longmapsto (STCal\ Source\ Target)\ (TargetTerm\ T2')$
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 $\exists\, Q'.\ TargetTerm\ T2 \longmapsto (STCal\ Source\ Target)\ Q' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
next
case (*trans P Q R*)
assume $P \longmapsto (STCal\ Source\ Target)\ P'$
and $\bigwedge P'.\ P \longmapsto (STCal\ Source\ Target)\ P'$
$\Longrightarrow \exists\, Q'.\ Q \longmapsto (STCal\ Source\ Target)\ Q' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
from *this* **obtain** Q' **where** $D1$: $Q \longmapsto (STCal\ Source\ Target)\ Q'$ **and** $D2$: $P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto (STCal\ Source\ Target)\ Q'$
$\Longrightarrow \exists\, R'.\ R \longmapsto (STCal\ Source\ Target)\ R' \wedge Q' \sim[\![\cdot]\!]T{<}TRel{>}\ R'$
with *D1* **obtain** R' **where** $D3$: $R \longmapsto (STCal\ Source\ Target)\ R'$ **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** $\exists\, R'.\ R \longmapsto (STCal\ Source\ Target)\ R' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ R'$
by *blast*
qed
next
fix $P\ Q\ Q'$
assume $P \sim[\![\cdot]\!]T{<}TRel{>}\ Q$ **and** $Q \longmapsto (STCal\ Source\ Target)\ Q'$
thus $\exists\, P'.\ P \longmapsto (STCal\ Source\ Target)\ P' \wedge P' \sim[\![\cdot]\!]T{<}TRel{>}\ Q'$

```
proof (induct arbitrary: Q')
  case (encR S)
  assume TargetTerm (⟦S⟧) ⟼(STCal Source Target) Q'
  from this obtain T where E1: ⟦S⟧ ⟼Target T and E2: T ∈T Q'
    by (auto simp add: STCal-step(2))
  from E1 oc obtain S' where E3: S ⟼Source S' and E4: (⟦S'⟧, T) ∈ TRel
    by blast
  from E3 have SourceTerm S ⟼(STCal Source Target) (SourceTerm S')
    by (simp add: STCal-step(1))
  moreover have SourceTerm S' ∼⟦·⟧T<TRel> Q'
  proof −
    have SourceTerm S' ∼⟦·⟧T<TRel> TargetTerm (⟦S'⟧)
      by (rule indRelTEQ.encR)
    moreover from E2 E4 have TargetTerm (⟦S'⟧) ∼⟦·⟧T<TRel> Q'
      by (simp add: indRelTEQ.target)
    ultimately show SourceTerm S' ∼⟦·⟧T<TRel> Q'
      by (rule indRelTEQ.trans)
  qed
  ultimately show ∃ P'. SourceTerm S ⟼(STCal Source Target) P' ∧ P' ∼⟦·⟧T<TRel> Q'
    by blast
next
  case (encL S)
  assume SourceTerm S ⟼(STCal Source Target) Q'
  from this obtain S' where F1: S ⟼Source S' and F2: S' ∈S Q'
    by (auto simp add: STCal-step(1))
  from F1 oc obtain T where F3: ⟦S⟧ ⟼Target T and F4: (⟦S'⟧, T) ∈ TRel
    by blast
  from F3 have TargetTerm (⟦S⟧) ⟼(STCal Source Target) (TargetTerm T)
    by (simp add: STCal-step(2))
  moreover have TargetTerm T ∼⟦·⟧T<TRel> Q'
  proof −
    from F4 eqT have (T, ⟦S'⟧) ∈ TRel
        unfolding equiv-def sym-def
      by blast
    hence TargetTerm T ∼⟦·⟧T<TRel> TargetTerm (⟦S'⟧)
      by (rule indRelTEQ.target)
    moreover from F2 have TargetTerm (⟦S'⟧) ∼⟦·⟧T<TRel> Q'
      by (simp add: indRelTEQ.encL)
    ultimately show TargetTerm T ∼⟦·⟧T<TRel> Q'
      by (rule indRelTEQ.trans)
  qed
  ultimately show ∃ P'. TargetTerm (⟦S⟧) ⟼(STCal Source Target) P' ∧ P' ∼⟦·⟧T<TRel> Q'
    by blast
next
  case (target T1 T2)
  assume TargetTerm T2 ⟼(STCal Source Target) Q'
  from this obtain T2' where G1: T2 ⟼Target T2' and G2: T2' ∈T Q'
    by (auto simp add: STCal-step(2))
  assume (T1, T2) ∈ TRel
  with G1 bisimT obtain T1' where G3: T1 ⟼Target T1' and G4: (T1', T2') ∈ TRel
    by blast
  from G3 have TargetTerm T1 ⟼(STCal Source Target) (TargetTerm T1')
    by (simp add: STCal-step(2))
  moreover from G2 G4 have TargetTerm T1' ∼⟦·⟧T<TRel> Q'
    by (simp add: indRelTEQ.target)
  ultimately show ∃ P'. TargetTerm T1 ⟼(STCal Source Target) P' ∧ P' ∼⟦·⟧T<TRel> Q'
    by blast
next
  case (trans P Q R R')
  assume R ⟼(STCal Source Target) R'
     and ⋀R'. R ⟼(STCal Source Target) R'
```

$\Longrightarrow \exists\, Q'.\ Q \longmapsto(\mathit{STCal\ Source\ Target})\ Q' \wedge Q' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ R'$
from *this* **obtain** Q' **where** *H1*: $Q \longmapsto(\mathit{STCal\ Source\ Target})\ Q'$ **and** *H2*: $Q' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ R'$
by *blast*
assume $\bigwedge Q'.\ Q \longmapsto(\mathit{STCal\ Source\ Target})\ Q'$
$\Longrightarrow \exists\, P'.\ P \longmapsto(\mathit{STCal\ Source\ Target})\ P' \wedge P' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ Q'$
with *H1* **obtain** P' **where** *H3*: $P \longmapsto(\mathit{STCal\ Source\ Target})\ P'$ **and** *H4*: $P' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ Q'$
by *blast*
from *H4 H2* **have** $P' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ R'$
by (*rule indRelTEQ.trans*)
with *H3* **show** $\exists\, P'.\ P \longmapsto(\mathit{STCal\ Source\ Target})\ P' \wedge P' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ R'$
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 $\mathit{SourceTerm}\ S \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ ([\![S]\!])$
by (*rule indRelTEQ.encR*)
moreover assume $S \longmapsto\mathit{Source}\ S'$
hence $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})\ (\mathit{SourceTerm}\ S')$
by (*simp add*: *STCal-step*(*1*))
ultimately obtain Q' **where** *I1*: $\mathit{TargetTerm}\ ([\![S]\!]) \longmapsto(\mathit{STCal\ Source\ Target})\ Q'$
and *I2*: $\mathit{SourceTerm}\ S' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ Q'$
using *bisim*
by *blast*
from *I1* **obtain** T **where** *I3*: $[\![S]\!] \longmapsto\mathit{Target}\ T$ **and** *I4*: $T \in T\ Q'$
by (*auto simp add*: *STCal-step*(*2*))
from *eqT* **have** $\mathit{TRel}^* = \mathit{TRel}$
using *reflcl-trancl*[*of TRel*] *trancl-id*[*of TRel*]
unfolding *equiv-def refl-on-def*
by *auto*
with *I2 I4* **have** $([\![S']\!],\ T) \in \mathit{TRel}$
using *indRelTEQ-to-TRel*(*2*)[**where** *TRel*=*TRel*]
trans-closure-of-TRel-refl-cond[**where** *TRel*=*TRel*]
by *simp*
with *I3* **show** $\exists\, T.\ [\![S]\!] \longmapsto\mathit{Target}\ T \wedge ([\![S']\!],\ T) \in \mathit{TRel}$
by *blast*
next
fix $S\ T$
have $\mathit{SourceTerm}\ S \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ ([\![S]\!])$
by (*rule indRelTEQ.encR*)
moreover assume $[\![S]\!] \longmapsto\mathit{Target}\ T$
hence $\mathit{TargetTerm}\ ([\![S]\!]) \longmapsto(\mathit{STCal\ Source\ Target})\ (\mathit{TargetTerm}\ T)$
by (*simp add*: *STCal-step*(*2*))
ultimately obtain Q' **where** *J1*: $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal\ Source\ Target})\ Q'$
and *J2*: $Q' \sim[\![\cdot]\!]T{<}\mathit{TRel}{>}\ \mathit{TargetTerm}\ T$
using *bisim*
by *blast*
from *J1* **obtain** S' **where** *J3*: $S \longmapsto\mathit{Source}\ S'$ **and** *J4*: $S' \in S\ Q'$
by (*auto simp add*: *STCal-step*(*1*))
from *eqT* **have** $\mathit{TRel}^* = \mathit{TRel}$
using *reflcl-trancl*[*of TRel*] *trancl-id*[*of TRel*]
unfolding *equiv-def refl-on-def*
by *auto*
with *J2 J4* **have** $([\![S']\!],\ T) \in \mathit{TRel}$
using *indRelTEQ-to-TRel*(*2*)[**where** *TRel*=*TRel*]
trans-closure-of-TRel-refl-cond[**where** *TRel*=*TRel*]
by *blast*
with *J3* **show** $\exists\, S'.\ S \longmapsto\mathit{Source}\ S' \wedge ([\![S']\!],\ T) \in \mathit{TRel}$

    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* ∧ *strong-reduction-bisimulation TRel Target*
    by *simp*
qed

lemma (**in** *encoding*) *SOC-wrt-equivalence-iff-strong-reduction-bisimulation*:
  fixes *TRel* :: (*'procT* × *'procT*) *set*
  assumes *eqT*: *equivalence TRel*
  shows (*strongly-operational-corresponding TRel* ∧ *strong-reduction-bisimulation TRel Target*)
      ⟷ (∃ *Rel*.
      (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
      ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
      ∧ *trans Rel* ∧ *strong-reduction-bisimulation Rel* (*STCal Source 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 ∀ *S*. *SourceTerm S* ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S*⟧) ∧ *TargetTerm* (⟦*S*⟧) ∼⟦·⟧*T*<*TRel*> *SourceTerm S*
    by (*simp add*: *indRelTEQ.encR indRelTEQ.encL*)
  moreover from *rt* **have** *TRel* = {(*T1*, *T2*). *TargetTerm T1* ∼⟦·⟧*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 *strong-reduction-bisimulation* (*indRelTEQ TRel*) (*STCal Source Target*)
    using *SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation*[**where** *TRel=TRel*]
  by *blast*
  ultimately
  show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
    ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} ∧ *trans Rel*
    ∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
  by *blast*
next
  assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
          ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
      ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} ∧ *trans Rel*
      ∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
  from *this* **obtain** *Rel* **where** *A1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
                  ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
   and *A2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} **and** *A3*: *trans Rel*
   and *A4*: *strong-reduction-bisimulation Rel* (*STCal Source Target*)
  by *blast*

have *strongly-operational-corresponding TRel*
proof *auto*
  fix $S\ S'$
  from *A1* **have** $(\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ (\llbracket S \rrbracket)) \in \mathit{Rel}$
    by *simp*
  moreover assume $S \longmapsto \mathit{Source}\ S'$
  hence $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ (\mathit{SourceTerm}\ S')$
    by (*simp add*: *STCal-step*(*1*))
  ultimately obtain Q' **where** *B1*: $\mathit{TargetTerm}\ (\llbracket S \rrbracket) \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ Q'$
                        and *B2*: $(\mathit{SourceTerm}\ S',\ Q') \in \mathit{Rel}$
      using *A4*
    by *blast*
  from *B1* **obtain** T **where** *B3*: $\llbracket S \rrbracket \longmapsto \mathit{Target}\ T$ **and** *B4*: $T \in T\ Q'$
    by (*auto simp add*: *STCal-step*(*2*))
  from *A1* **have** $(\mathit{TargetTerm}\ (\llbracket S' \rrbracket),\ \mathit{SourceTerm}\ S') \in \mathit{Rel}$
    by *simp*
  with *B2 A3* **have** $(\mathit{TargetTerm}\ (\llbracket S' \rrbracket),\ Q') \in \mathit{Rel}$
      unfolding *trans-def*
    by *blast*
  with *B4 A2* **have** $(\llbracket S' \rrbracket,\ T) \in \mathit{TRel}$
    by *simp*
  with *B3* **show** $\exists\, T.\ \llbracket S \rrbracket \longmapsto \mathit{Target}\ T \wedge (\llbracket S' \rrbracket,\ T) \in \mathit{TRel}$
    by *blast*
next
  fix $S\ T$
  from *A1* **have** $(\mathit{SourceTerm}\ S,\ \mathit{TargetTerm}\ (\llbracket S \rrbracket)) \in \mathit{Rel}$
    by *simp*
  moreover assume $\llbracket S \rrbracket \longmapsto \mathit{Target}\ T$
  hence $\mathit{TargetTerm}\ (\llbracket S \rrbracket) \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ (\mathit{TargetTerm}\ T)$
    by (*simp add*: *STCal-step*(*2*))
  ultimately obtain P' **where** *C1*: $\mathit{SourceTerm}\ S \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ P'$
                        and *C2*: $(P',\ \mathit{TargetTerm}\ T) \in \mathit{Rel}$
      using *A4*
    by *blast*
  from *C1* **obtain** S' **where** *C3*: $S \longmapsto \mathit{Source}\ S'$ **and** *C4*: $S' \in S\ P'$
    by (*auto simp add*: *STCal-step*(*1*))
  from *A1 C4* **have** $(\mathit{TargetTerm}\ (\llbracket S' \rrbracket),\ P') \in \mathit{Rel}$
    by *simp*
  from *A3 this C2* **have** $(\mathit{TargetTerm}\ (\llbracket S' \rrbracket),\ \mathit{TargetTerm}\ T) \in \mathit{Rel}$
      unfolding *trans-def*
    by *blast*
  with *A2* **have** $(\llbracket S' \rrbracket,\ T) \in \mathit{TRel}$
    by *simp*
  with *C3* **show** $\exists\, S'.\ S \longmapsto \mathit{Source}\ S' \wedge (\llbracket S' \rrbracket,\ T) \in \mathit{TRel}$
    by *blast*
qed
moreover have *strong-reduction-bisimulation TRel Target*
proof *auto*
  fix $\mathit{TP}\ \mathit{TQ}\ \mathit{TP}'$
  assume $(\mathit{TP},\ \mathit{TQ}) \in \mathit{TRel}$
  with *A2* **have** $(\mathit{TargetTerm}\ \mathit{TP},\ \mathit{TargetTerm}\ \mathit{TQ}) \in \mathit{Rel}$
    by *simp*
  moreover assume $\mathit{TP} \longmapsto \mathit{Target}\ \mathit{TP}'$
  hence $\mathit{TargetTerm}\ \mathit{TP} \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ (\mathit{TargetTerm}\ \mathit{TP}')$
    by (*simp add*: *STCal-step*(*2*))
  ultimately obtain Q' **where** *D1*: $\mathit{TargetTerm}\ \mathit{TQ} \longmapsto(\mathit{STCal}\ \mathit{Source}\ \mathit{Target})\ Q'$
                        and *D2*: $(\mathit{TargetTerm}\ \mathit{TP}',\ Q') \in \mathit{Rel}$
      using *A4*
    by *blast*
  from *D1* **obtain** TQ' **where** *D3*: $\mathit{TQ} \longmapsto \mathit{Target}\ \mathit{TQ}'$ **and** *D4*: $\mathit{TQ}' \in T\ Q'$
    by (*auto simp add*: *STCal-step*(*2*))

from *A2 D2 D4* **have** $(TP', TQ') \in TRel$
by *simp*
with *D3* **show** $\exists TQ'. TQ \longmapsto Target\ TQ' \wedge (TP', TQ') \in TRel$
by *blast*
next
fix *TP TQ TQ'*
assume $(TP, TQ) \in TRel$
with *A2* **have** $(TargetTerm\ TP, TargetTerm\ TQ) \in Rel$
by *simp*
moreover assume $TQ \longmapsto Target\ TQ'$
hence $TargetTerm\ TQ \longmapsto (STCal\ Source\ Target)\ (TargetTerm\ TQ')$
by (*simp add*: *STCal-step*(*2*))
ultimately obtain P' **where** *E1*: $TargetTerm\ TP \longmapsto (STCal\ Source\ Target)\ P'$
and *E2*: $(P', TargetTerm\ TQ') \in Rel$
using *A4*
by *blast*
from *E1* **obtain** TP' **where** *E3*: $TP \longmapsto Target\ TP'$ **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** $\exists TP'. TP \longmapsto Target\ TP' \wedge (TP', TQ') \in TRel$
by *blast*
qed
ultimately
show *strongly-operational-corresponding TRel* $\wedge$ *strong-reduction-bisimulation TRel 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 (enc S1, enc S2) in TRel.

abbreviation (**in** *encoding*) *fully-abstract*
:: $('procS \times 'procS)\ set \Rightarrow ('procT \times 'procT)\ set \Rightarrow bool$
where
$fully\text{-}abstract\ SRel\ TRel \equiv \forall S1\ S2.\ (S1, S2) \in SRel \longleftrightarrow ([\![S1]\!], [\![S2]\!]) \in TRel$

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*: $\forall S1\ S2.\ [\![S1]\!] = [\![S2]\!] \longrightarrow S1 = S2$
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* :: $('procS \times 'procS)$ *set*
  assumes *injectivity*: $\forall S1\ S2.\ [\![S1]\!] = [\![S2]\!] \longrightarrow S1 = S2$
  shows $\exists RelT.$ *fully-abstract RelS RelT*
proof −
  define *RelT* **where** $RelT = \{(T1, T2).\ \exists S1\ S2.\ (S1, S2) \in RelS \wedge T1 = [\![S1]\!] \wedge T2 = [\![S2]\!]\}$
  with *injectivity* **have** *fully-abstract RelS RelT*
    by *blast*
  thus $\exists RelT.$ *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* :: $('procS \times 'procS)$ *set*
  assumes *encRelS*: $\forall S1\ S2.\ [\![S1]\!] = [\![S2]\!] \longrightarrow (S1, S2) \in RelS$
      and *transS*: *trans RelS*
  shows $\exists RelT.$ *fully-abstract RelS RelT*
proof −
  define *RelT* **where** $RelT = \{(T1, T2).\ \exists S1\ S2.\ (S1, S2) \in RelS \wedge T1 = [\![S1]\!] \wedge T2 = [\![S2]\!]\}$
  have *fully-abstract RelS RelT*
  proof *auto*
    fix *S1 S2*
    assume $(S1, S2) \in RelS$
    with *RelT-def* **show** $([\![S1]\!], [\![S2]\!]) \in RelT$
      by *blast*
  next
    fix *S1 S2*
    assume $([\![S1]\!], [\![S2]\!]) \in RelT$
    with *RelT-def* **obtain** $S1'\ S2'$ **where** *A1*: $(S1', S2') \in RelS$ **and** *A2*: $[\![S1]\!] = [\![S1']\!]$
                         and *A3*: $[\![S2]\!] = [\![S2']\!]$
      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 $\exists RelT.$ *fully-abstract RelS RelT*
    by *blast*
qed

lemma (**in** *encoding*) *fully-abstract-wrt-trans-closure-of-source-relation*:
  fixes *RelS* :: $('procS \times 'procS)$ *set*
  assumes *encRelS*: $\forall S1\ S2.\ [\![S1]\!] = [\![S2]\!] \longrightarrow (S1, S2) \in RelS^+$
  shows $\exists RelT.$ *fully-abstract* $(RelS^+)$ *RelT*

using *encRelS trans-trancl*[*of RelS*]
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* × '*procT*) *set*
shows ∃ *RelS*. *fully-abstract RelS RelT*
proof −
define *RelS* **where** *RelS* = {(*S1*, *S2*). (⟦*S1*⟧, ⟦*S2*⟧) ∈ *RelT*}
hence *fully-abstract RelS RelT*
by *simp*
thus ∃ *RelS*. *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* × ('*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 *srel*: *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *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*
proof *auto*
fix *S1 S2*
define *Rel'* **where** *Rel'* = *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*}
hence (*TargetTerm* (⟦*S1*⟧), *SourceTerm S1*) ∈ *Rel'*
by *simp*
moreover assume (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
with *fullAbs* **have** (*S1*, *S2*) ∈ *SRel*
by *simp*
with *srel Rel'-def* **have** (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel'*
by *simp*
moreover from *encR Rel'-def* **have** (*SourceTerm S2*, *TargetTerm* (⟦*S2*⟧)) ∈ *Rel'*
by *simp*
ultimately show (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
using *trans Rel'-def*
unfolding *trans-def*
by *blast*
next
fix *S1 S2*
define *Rel'* **where** *Rel'* = *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*}
from *encR Rel'-def* **have** (*SourceTerm S1*, *TargetTerm* (⟦*S1*⟧)) ∈ *Rel'*
by *simp*
moreover assume (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
with *Rel'-def* **have** (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel'*
by *simp*
moreover from *Rel'-def* **have** (*TargetTerm* (⟦*S2*⟧), *SourceTerm S2*) ∈ *Rel'*

    by *simp*
  ultimately have (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
      using *trans Rel′-def*
      unfolding *trans-def*
    by *blast*
  with *srel* **have** (*S1*, *S2*) ∈ *SRel*
    by *simp*
  with *fullAbs* **show** (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
    by *simp*
qed

lemma (**in** *encoding*) *full-abstraction-and-trans-relation-contains-TRel-impl-SRel*:
  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 *trel*:   ∀ *S1 S2*. (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel* ⟷ (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
      and *trans*:   *trans* (*Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*})
  shows *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
proof *auto*
  fix *S1 S2*
  define *Rel′* **where** *Rel′* = *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*}
  from *encR Rel′-def* **have** (*SourceTerm S1*, *TargetTerm* (⟦*S1*⟧)) ∈ *Rel′*
    by *simp*
  moreover assume (*S1*, *S2*) ∈ *SRel*
  with *fullAbs* **have** (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
    by *simp*
  with *trel Rel′-def* **have** (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel′*
    by *simp*
  moreover from *Rel′-def* **have** (*TargetTerm* (⟦*S2*⟧), *SourceTerm S2*) ∈ *Rel′*
    by *simp*
  ultimately show (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
      using *trans Rel′-def*
      unfolding *trans-def*
    by *blast*
next
  fix *S1 S2*
  define *Rel′* **where** *Rel′* = *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*}
  hence (*TargetTerm* (⟦*S1*⟧), *SourceTerm S1*) ∈ *Rel′*
    by *simp*
  moreover assume (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
  with *Rel′-def* **have** (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel′*
    by *simp*
  moreover from *encR Rel′-def* **have** (*SourceTerm S2*, *TargetTerm* (⟦*S2*⟧)) ∈ *Rel′*
    by *simp*
  ultimately have (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
      using *trans Rel′-def*
      unfolding *trans-def*
    by *blast*
  with *trel* **have** (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
    by *simp*
  with *fullAbs* **show** (*S1*, *S2*) ∈ *SRel*
    by *simp*
qed

lemma (**in** *encoding*) *full-abstraction-impl-trans-relation-contains-SRel-iff-TRel*:
  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 *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** $T = [\![S]\!]$
by *blast*
moreover from *reflS* **have** $(S, S) \in SRel$
unfolding *refl-on-def*
by *simp*
with *fullAbs* **have** $([\![S]\!], [\![S]\!]) \in TRel$
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 $sym\ SRel \longleftrightarrow sym\ \{(T1, T2).\ \exists S1\ S2.\ T1 = [\![S1]\!] \wedge T2 = [\![S2]\!] \wedge (T1, T2) \in TRel\}$
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*: $\forall T.\ \exists S.\ T = [\![S]\!]$
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 $trans\ SRel \longleftrightarrow trans\ \{(T1, T2).\ \exists S1\ S2.\ T1 = [\![S1]\!] \wedge T2 = [\![S2]\!] \wedge (T1, T2) \in TRel\}$
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*: $\forall T.\ \exists S.\ T = [\![S]\!]$
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 $TRel :: ('procT \times 'procT)\ set$
and $Pred :: ('procS, 'procT)\ Proc \Rightarrow bool$

assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-pred Pred*
shows *rel-preserves-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S* *P* ∧ *SQ* ∈*S* *Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
⟷ *rel-preserves-pred* {(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T* *P* ∧ ⟦*SQ*⟧ ∈*T* *Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*} *Pred*
proof
assume *presS*: *rel-preserves-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S* *P* ∧ *SQ* ∈*S* *Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
show *rel-preserves-pred* {(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T* *P* ∧ ⟦*SQ*⟧ ∈*T* *Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*} *Pred*
proof *clarify*
fix *SP SQ*
assume *Pred* (*TargetTerm* (⟦*SP*⟧))
with *encP* **have** *Pred* (*SourceTerm SP*)
by *simp*
moreover assume (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
with *fullAbs* **have** (*SP*, *SQ*) ∈ *SRel*
by *simp*
ultimately have *Pred* (*SourceTerm SQ*)
using *presS*
by *blast*
with *encP* **show** *Pred* (*TargetTerm* (⟦*SQ*⟧))
by *simp*
qed
next
assume *presT*:
rel-preserves-pred {(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T* *P* ∧ ⟦*SQ*⟧ ∈*T* *Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*} *Pred*
show *rel-preserves-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S* *P* ∧ *SQ* ∈*S* *Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
proof *clarify*
fix *SP SQ*
assume *Pred* (*SourceTerm SP*)
with *encP* **have** *Pred* (*TargetTerm* (⟦*SP*⟧))
by *simp*
moreover assume (*SP*, *SQ*) ∈ *SRel*
with *fullAbs* **have** (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
by *simp*
ultimately have *Pred* (*TargetTerm* (⟦*SQ*⟧))
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* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *set*
and *Pred* :: (*'procS*, *'procT*) *Proc* ⇒ *'b* ⇒ *bool*
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-binary-pred Pred*
shows *rel-preserves-binary-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S* *P* ∧ *SQ* ∈*S* *Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
⟷ *rel-preserves-binary-pred*
{(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T* *P* ∧ ⟦*SQ*⟧ ∈*T* *Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*} *Pred*
proof
assume *presS*:
rel-preserves-binary-pred {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S* *P* ∧ *SQ* ∈*S* *Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
show *rel-preserves-binary-pred*
{(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T* *P* ∧ ⟦*SQ*⟧ ∈*T* *Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*} *Pred*
proof *clarify*
fix *x SP SQ*
assume *Pred* (*TargetTerm* (⟦*SP*⟧)) *x*
with *encP* **have** *Pred* (*SourceTerm SP*) *x*
by *simp*
moreover assume (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *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* ($[\![SQ]\!]$)) *x*
by *simp*
qed
next
assume *presT*:
rel-preserves-binary-pred $\{(P, Q). \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *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* ($[\![SP]\!]$)) *x*
by *simp*
moreover assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
ultimately have *Pred* (*TargetTerm* ($[\![SQ]\!]$)) *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* $\{(P, Q). \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *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* $\{(P, Q). \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *Pred*
proof *clarify*
fix *SP SQ*
assume *Pred* (*TargetTerm* ($[\![SQ]\!]$))
with *encP* **have** *Pred* (*SourceTerm SQ*)
by *simp*
moreover assume $([\![SP]\!], [\![SQ]\!]) \in TRel$
with *fullAbs* **have** $(SP, SQ) \in SRel$
by *simp*
ultimately have *Pred* (*SourceTerm SP*)
using *reflS*
by *blast*
with *encP* **show** *Pred* (*TargetTerm* ($[\![SP]\!]$))
by *simp*
qed
next
assume *reflT*:
rel-reflects-pred $\{(P, Q). \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *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* ($[\![SQ]\!]$))

by *simp*
moreover assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
ultimately have $Pred\ (TargetTerm\ ([\![SP]\!]))$
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*
$\{(P, Q).\ \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *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*
$\{(P, Q).\ \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *Pred*
proof *clarify*
fix *x SP SQ*
assume $Pred\ (TargetTerm\ ([\![SQ]\!]))\ x$
with *encP* **have** $Pred\ (SourceTerm\ SQ)\ x$
by *simp*
moreover assume $([\![SP]\!], [\![SQ]\!]) \in TRel$
with *fullAbs* **have** $(SP, SQ) \in SRel$
by *simp*
ultimately have $Pred\ (SourceTerm\ SP)\ x$
using *reflS*
by *blast*
with *encP* **show** $Pred\ (TargetTerm\ ([\![SP]\!]))\ x$
by *simp*
qed
next
assume *reflT*:
rel-reflects-binary-pred $\{(P, Q).\ \exists SP\ SQ.\ [\![SP]\!] \in T\ P \wedge [\![SQ]\!] \in T\ Q \wedge ([\![SP]\!], [\![SQ]\!]) \in TRel\}$ *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** $Pred\ (TargetTerm\ ([\![SQ]\!]))\ x$
by *simp*
moreover assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
ultimately have $Pred\ (TargetTerm\ ([\![SP]\!]))\ x$
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* :: ($'procS$ × $'procS$) *set*
and *TRel* :: ($'procT$ × $'procT$) *set*
and *Pred* :: ($'procS$, $'procT$) *Proc* ⇒ *bool*
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-pred Pred*
shows *rel-respects-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S P* ∧ *SQ* ∈*S Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
⟷ *rel-respects-pred* {(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T P* ∧ ⟦*SQ*⟧ ∈*T Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *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* :: ($'procS$ × $'procS$) *set*
and *TRel* :: ($'procT$ × $'procT$) *set*
and *Pred* :: ($'procS$, $'procT$) *Proc* ⇒ $'b$ ⇒ *bool*
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-binary-pred Pred*
shows *rel-respects-binary-pred* {(*P*, *Q*). ∃ *SP SQ*. *SP* ∈*S P* ∧ *SQ* ∈*S Q* ∧ (*SP*, *SQ*) ∈ *SRel*} *Pred*
⟷ *rel-respects-binary-pred*
{(*P*, *Q*). ∃ *SP SQ*. ⟦*SP*⟧ ∈*T P* ∧ ⟦*SQ*⟧ ∈*T Q* ∧ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *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* :: (($'procS$, $'procT$) *Proc* × ($'procS$, $'procT$) *Proc*) *set*
assumes *enc*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
and *trans*: *trans Rel*
shows *fully-abstract* {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
{(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
proof *auto*
fix *S1 S2*
assume (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
with *enc trans* **show** (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
unfolding *trans-def*
by *blast*
next
fix *S1 S2*
assume (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel*
with *enc trans* **show** (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*
unfolding *trans-def*
by *blast*
qed

lemma (**in** *encoding*) *source-target-relation-impl-full-abstraction-wrt-trans-closures*:
fixes *Rel* :: (($'procS$, $'procT$) *Proc* × ($'procS$, $'procT$) *Proc*) *set*
assumes *enc*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
shows *fully-abstract* {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ Rel^+}
{(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ 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** $(\mathit{TargetTerm}\ (\llbracket S2 \rrbracket),\ \mathit{SourceTerm}\ S2) \in \mathit{Rel}$
by *simp*
ultimately have $(\mathit{SourceTerm}\ S1,\ \mathit{SourceTerm}\ S2) \in \mathit{Rel}$
using *trans*
by *blast*
with *srel* **show** $(S1,\ S2) \in \mathit{SRel}$
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 $\mathit{SRel} :: ('\mathit{procS} \times '\mathit{procS})\ \mathit{set}$
and $\mathit{TRel} :: ('\mathit{procT} \times '\mathit{procT})\ \mathit{set}$
and $P\ Q :: ('\mathit{procS},\ '\mathit{procT})\ \mathit{Proc}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *rel*: $P \lesssim\llbracket\cdot\rrbracket R\text{<}\mathit{SRel},\mathit{TRel}\text{>}\ Q$
shows $\forall\, SP\ SQ.\ SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (\llbracket SP \rrbracket,\ \llbracket SQ \rrbracket) \in \mathit{TRel}^+$
and $\forall\, SP\ TQ.\ SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\llbracket SP \rrbracket,\ TQ) \in \mathit{TRel}^*$
proof −
have *fullAbsT*: $\forall\, S1\ S2.\ (S1,\ S2) \in \mathit{SRel}^+ \longrightarrow (\llbracket S1 \rrbracket,\ \llbracket S2 \rrbracket) \in \mathit{TRel}^+$
proof *clarify*
fix *S1 S2*
assume $(S1,\ S2) \in \mathit{SRel}^+$
thus $(\llbracket S1 \rrbracket,\ \llbracket S2 \rrbracket) \in \mathit{TRel}^+$
proof *induct*
fix *S2*
assume $(S1,\ S2) \in \mathit{SRel}$
with *fullAbs* **show** $(\llbracket S1 \rrbracket,\ \llbracket S2 \rrbracket) \in \mathit{TRel}^+$
by *simp*
next
case (*step S2 S3*)
assume $(\llbracket S1 \rrbracket,\ \llbracket S2 \rrbracket) \in \mathit{TRel}^+$
moreover assume $(S2,\ S3) \in \mathit{SRel}$
with *fullAbs* **have** $(\llbracket S2 \rrbracket,\ \llbracket S3 \rrbracket) \in \mathit{TRel}^+$
by *simp*
ultimately show $(\llbracket S1 \rrbracket,\ \llbracket S3 \rrbracket) \in \mathit{TRel}^+$
by *simp*
qed
qed
with *rel* **show** $\forall\, SP\ SQ.\ SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (\llbracket SP \rrbracket,\ \llbracket SQ \rrbracket) \in \mathit{TRel}^+$
using *indRelRSTPO-to-SRel-and-TRel*(*1*)[**where** *SRel*=*SRel* **and** *TRel*=*TRel*]
by *simp*
show $\forall\, SP\ TQ.\ SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\llbracket SP \rrbracket,\ TQ) \in \mathit{TRel}^*$
proof *clarify*
fix *SP TQ*
assume $SP \in S\ P$ **and** $TQ \in T\ Q$
with *rel* **obtain** *S* **where** *A1*: $(SP,\ S) \in \mathit{SRel}^*$
and *A2*: $(\llbracket S \rrbracket,\ TQ) \in \mathit{TRel}^*$
using *indRelRSTPO-to-SRel-and-TRel*(*2*)[**where** *SRel*=*SRel* **and** *TRel*=*TRel*]
by *blast*
from *A1* **have** $SP = S \vee (SP,\ S) \in \mathit{SRel}^+$
using *rtrancl-eq-or-trancl*[*of SP S SRel*]
by *blast*
with *fullAbsT* **have** $(\llbracket SP \rrbracket,\ \llbracket S \rrbracket) \in \mathit{TRel}^*$
by *fast*
from *this A2* **show** $(\llbracket SP \rrbracket,\ TQ) \in \mathit{TRel}^*$
by *simp*
qed

qed

lemma (**in** *encoding*) *full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel*:
  fixes *SRel* :: (*'procS* × *'procS*) *set*
    and *TRel* :: (*'procT* × *'procT*) *set*
    and *P Q* :: (*'procS*, *'procT*) *Proc*
  assumes *fA*: *fully-abstract SRel TRel*
      and *transT*: *trans TRel*
      and *reflS*: *refl SRel*
      and *rel*: *P* ∼⟦·⟧<*SRel*,*TRel*> *Q*
  shows ∀ *SP SQ*. *SP* ∈*S P* ∧ *SQ* ∈*S Q* ⟶ (*SP*, *SQ*) ∈ *SRel*
    and ∀ *SP SQ*. *SP* ∈*S P* ∧ *SQ* ∈*S Q* ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
    and ∀ *SP TQ*. *SP* ∈*S P* ∧ *TQ* ∈*T Q* ⟶ (⟦*SP*⟧, *TQ*) ∈ *TRel*
    and ∀ *TP SQ*. *TP* ∈*T P* ∧ *SQ* ∈*S Q* ⟶ (*TP*, ⟦*SQ*⟧) ∈ *TRel*
    and ∀ *TP TQ*. *TP* ∈*T P* ∧ *TQ* ∈*T Q* ⟶ (*TP*, *TQ*) ∈ *TRel*
      using *rel*
proof *induct*
  case (*encR S*)
  show ∀ *SP SQ*. *SP* ∈*S SourceTerm S* ∧ *SQ* ∈*S TargetTerm* (⟦*S*⟧) ⟶ (*SP*, *SQ*) ∈ *SRel*
   and ∀ *SP SQ*. *SP* ∈*S SourceTerm S* ∧ *SQ* ∈*S TargetTerm* (⟦*S*⟧) ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
   and ∀ *TP SQ*. *TP* ∈*T SourceTerm S* ∧ *SQ* ∈*S TargetTerm* (⟦*S*⟧) ⟶ (*TP*, ⟦*SQ*⟧) ∈ *TRel*
   and ∀ *TP TQ*. *TP* ∈*T SourceTerm S* ∧ *TQ* ∈*T TargetTerm* (⟦*S*⟧) ⟶ (*TP*, *TQ*) ∈ *TRel*
    by *simp+*
  from *reflS fA* **show** ∀ *SP TQ*. *SP* ∈*S SourceTerm S* ∧ *TQ* ∈*T TargetTerm* (⟦*S*⟧) ⟶ (⟦*SP*⟧, *TQ*) ∈ *TRel*
      unfolding *refl-on-def*
    by *simp*
next
  case (*encL S*)
  show ∀ *SP SQ*. *SP* ∈*S TargetTerm* (⟦*S*⟧) ∧ *SQ* ∈*S SourceTerm S* ⟶ (*SP*, *SQ*) ∈ *SRel*
   and ∀ *SP SQ*. *SP* ∈*S TargetTerm* (⟦*S*⟧) ∧ *SQ* ∈*S SourceTerm S* ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
   and ∀ *SP TQ*. *SP* ∈*S TargetTerm* (⟦*S*⟧) ∧ *TQ* ∈*T SourceTerm S* ⟶ (⟦*SP*⟧, *TQ*) ∈ *TRel*
   and ∀ *TP TQ*. *TP* ∈*T TargetTerm* (⟦*S*⟧) ∧ *TQ* ∈*T SourceTerm S* ⟶ (*TP*, *TQ*) ∈ *TRel*
    by *simp+*
  with *reflS fA* **show** ∀ *TP SQ*. *TP* ∈*T TargetTerm* (⟦*S*⟧) ∧ *SQ* ∈*S SourceTerm S* ⟶ (*TP*, ⟦*SQ*⟧) ∈ *TRel*
      unfolding *refl-on-def*
    by *simp*
next
  case (*source S1 S2*)
  show ∀ *SP TQ*. *SP* ∈*S SourceTerm S1* ∧ *TQ* ∈*T SourceTerm S2* ⟶ (⟦*SP*⟧, *TQ*) ∈ *TRel*
   and ∀ *TP SQ*. *TP* ∈*T SourceTerm S1* ∧ *SQ* ∈*S SourceTerm S2* ⟶ (*TP*, ⟦*SQ*⟧) ∈ *TRel*
   and ∀ *TP TQ*. *TP* ∈*T SourceTerm S1* ∧ *TQ* ∈*T SourceTerm S2* ⟶ (*TP*, *TQ*) ∈ *TRel*
    by *simp+*
  assume (*S1*, *S2*) ∈ *SRel*
  thus ∀ *SP SQ*. *SP* ∈*S SourceTerm S1* ∧ *SQ* ∈*S SourceTerm S2* ⟶ (*SP*, *SQ*) ∈ *SRel*
    by *simp*
  with *fA* **show** ∀ *SP SQ*. *SP* ∈*S SourceTerm S1* ∧ *SQ* ∈*S SourceTerm S2* ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
    by *simp*
next
  case (*target T1 T2*)
  show ∀ *SP SQ*. *SP* ∈*S TargetTerm T1* ∧ *SQ* ∈*S TargetTerm T2* ⟶ (*SP*, *SQ*) ∈ *SRel*
   and ∀ *SP SQ*. *SP* ∈*S TargetTerm T1* ∧ *SQ* ∈*S TargetTerm T2* ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
   and ∀ *SP TQ*. *SP* ∈*S TargetTerm T1* ∧ *TQ* ∈*T TargetTerm T2* ⟶ (⟦*SP*⟧, *TQ*) ∈ *TRel*
   and ∀ *TP SQ*. *TP* ∈*T TargetTerm T1* ∧ *SQ* ∈*S TargetTerm T2* ⟶ (*TP*, ⟦*SQ*⟧) ∈ *TRel*
    by *simp+*
  assume (*T1*, *T2*) ∈ *TRel*
  thus ∀ *TP TQ*. *TP* ∈*T TargetTerm T1* ∧ *TQ* ∈*T TargetTerm T2* ⟶ (*TP*, *TQ*) ∈ *TRel*
    by *simp*
next
  case (*trans P Q R*)
  assume *A1*: ∀ *SP SQ*. *SP* ∈*S P* ∧ *SQ* ∈*S Q* ⟶ (⟦*SP*⟧, ⟦*SQ*⟧) ∈ *TRel*
    and *A2*: ∀ *SP TQ*. *SP* ∈*S P* ∧ *TQ* ∈*T Q* ⟶ (⟦*SP*⟧, *TQ*) ∈ *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 ∀ TP SR. TP ∈T P ∧ SR ∈S R ⟶ (TP, ⟦SR⟧) ∈ TRel
proof clarify
  fix TP SR
  assume A9: TP ∈T P and A10: SR ∈S R
  show (TP, ⟦SR⟧) ∈ TRel
  proof (cases Q)
    case (SourceTerm SQ)
    assume A11: SQ ∈S Q
    with A3 A9 have (TP, ⟦SQ⟧) ∈ TRel
      by blast
    moreover from A5 A10 A11 have (⟦SQ⟧, ⟦SR⟧) ∈ TRel
      by blast
    ultimately show (TP, ⟦SR⟧) ∈ TRel
        using transT
        unfolding trans-def
      by blast
  next
    case (TargetTerm TQ)
    assume A11: TQ ∈T Q
    with A4 A9 have (TP, TQ) ∈ TRel
      by blast
    moreover from A7 A10 A11 have (TQ, ⟦SR⟧) ∈ TRel
      by blast
    ultimately show (TP, ⟦SR⟧) ∈ TRel
        using transT
        unfolding trans-def
      by blast
  qed
qed
show ∀ TP TR. TP ∈T P ∧ TR ∈T R ⟶ (TP, TR) ∈ TRel
proof clarify
  fix TP TR
  assume A9: TP ∈T P and A10: TR ∈T R
  show (TP, TR) ∈ TRel
  proof (cases Q)
    case (SourceTerm SQ)
    assume A11: SQ ∈S Q
    with A3 A9 have (TP, ⟦SQ⟧) ∈ TRel
      by blast
    moreover from A6 A10 A11 have (⟦SQ⟧, TR) ∈ TRel
      by blast
    ultimately show (TP, TR) ∈ TRel
        using transT
        unfolding trans-def
      by blast
  next
    case (TargetTerm TQ)
    assume A11: TQ ∈T Q
    with A4 A9 have (TP, TQ) ∈ TRel
      by blast
    moreover from A8 A10 A11 have (TQ, TR) ∈ TRel
      by blast
    ultimately show (TP, TR) ∈ 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 × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
      and reflS:   refl SRel
      and transT:  trans TRel
  shows ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
                    ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
                ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
                ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                ∧ trans Rel
proof −
  have ∀ S. SourceTerm S ∼⟦·⟧<SRel,TRel> TargetTerm (⟦S⟧)
        ∧ TargetTerm (⟦S⟧) ∼⟦·⟧<SRel,TRel> 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 ∼⟦·⟧<SRel,TRel> SourceTerm S2}
  proof auto
    fix S1 S2
    assume (S1, S2) ∈ SRel
    thus SourceTerm S1 ∼⟦·⟧<SRel,TRel> SourceTerm S2
      by (rule indRelSTEQ.source[where SRel=SRel and TRel=TRel])
  next
    fix S1 S2
    assume SourceTerm S1 ∼⟦·⟧<SRel,TRel> SourceTerm S2
    with fullAbs reflS transT 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 have TRel =  {(T1, T2). TargetTerm T1 ∼⟦·⟧<SRel,TRel> TargetTerm T2}
  proof auto
    fix T1 T2
    assume (T1, T2) ∈ TRel
    thus TargetTerm T1 ∼⟦·⟧<SRel,TRel> TargetTerm T2
      by (rule indRelSTEQ.target[where SRel=SRel and TRel=TRel])
  next
    fix T1 T2
    assume TargetTerm T1 ∼⟦·⟧<SRel,TRel> TargetTerm T2
    with fullAbs reflS transT 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 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 × 'procS) set
    and TRel :: ('procT × 'procT) set
  shows (fully-abstract SRel TRel ∧ refl SRel ∧ trans TRel) =
         (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
                      ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
                ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
                ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                ∧ trans Rel)
proof (rule iffI)
  assume fully-abstract SRel TRel ∧ refl SRel ∧ trans TRel
  thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
                ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
                ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                ∧ trans Rel
      using full-abstraction-wrt-preorders-impl-trans-source-target-relation[where SRel=SRel
            and TRel=TRel]
    by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
                 ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
                ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
                ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
                ∧ trans Rel
  from this obtain Rel
    where A1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel
      and A2: SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
      and A3: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ 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 (⟦S⟧)) ∈ Rel
      by blast
    moreover from A1 have (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel
      by blast
    ultimately have (SourceTerm S, SourceTerm S) ∈ Rel
        using A4
        unfolding trans-def
      by blast
    with A2 show (S, S) ∈ SRel
      by blast
  qed
  moreover from A3 A4 have trans TRel
      unfolding trans-def
    by blast
  ultimately show fully-abstract SRel TRel ∧ refl SRel ∧ 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* × ('*procS*, '*procT*) *Proc*) *set*
  assumes *enc*: $\forall S$. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) ∈ *Rel*
    and *trans*: *trans* (*symcl Rel*)
  shows *fully-abstract* {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *symcl Rel*}
      {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *symcl Rel*}
proof *auto*
  fix *S1 S2*
  from *enc* **have** (*TargetTerm* ($[\![S1]\!]$), *SourceTerm S1*) ∈ *symcl Rel*
   by (*simp add*: *symcl-def*)
  moreover assume (*SourceTerm S1*, *SourceTerm S2*) ∈ *symcl Rel*
  moreover from *enc* **have** (*SourceTerm S2*, *TargetTerm* ($[\![S2]\!]$)) ∈ *symcl Rel*
   by (*simp add*: *symcl-def*)
  ultimately show (*TargetTerm* ($[\![S1]\!]$), *TargetTerm* ($[\![S2]\!]$)) ∈ *symcl Rel*
    using *trans*
    unfolding *trans-def*
   by *blast*
next
  fix *S1 S2*
  from *enc* **have** (*SourceTerm S1*, *TargetTerm* ($[\![S1]\!]$)) ∈ *symcl Rel*
   by (*simp add*: *symcl-def*)
  moreover assume (*TargetTerm* ($[\![S1]\!]$), *TargetTerm* ($[\![S2]\!]$)) ∈ *symcl Rel*
  moreover from *enc* **have** (*TargetTerm* ($[\![S2]\!]$), *SourceTerm S2*) ∈ *symcl Rel*
   by (*simp add*: *symcl-def*)
  ultimately show (*SourceTerm S1*, *SourceTerm S2*) ∈ *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* × '*procS*) *set*
   and *TRel* :: ('*procT* × '*procT*) *set*
  assumes *fullAbs*: *fully-abstract SRel TRel*
    and *reflT*: *refl TRel*
    and *symmT*: *sym TRel*
    and *transT*: *trans TRel*
  shows ∃ *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 −
  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 ∀ *S*. *SourceTerm S* $\lesssim[\![\cdot]\!]R<SRel,TRel>$ *TargetTerm* ($[\![S]\!]$)

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*)

```
  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
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)
      next
        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 $Q \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P$
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** $Q \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P$
by (*simp add*: *indRelRSTPO.target*)
qed
qed
qed
moreover
have $\forall P\ Q\ R.\ P \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q \wedge P \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R \wedge (Q,\ R) \notin (indRelRSTPO\ SRel\ TRel)$
$\longrightarrow Q \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P \vee R \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P$
proof *clarify*
fix *P Q R*
assume *A1*: $P \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ Q$ **and** *A2*: $P \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ R$
and *A3*: (*Q*, *R*) ∉ (*indRelRSTPO SRel TRel*) **and** *A4*: (*R*, *P*) ∉ (*indRelRSTPO SRel TRel*)
show $Q \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P$
proof (*cases P*)
case (*SourceTerm SP*)
assume *A5*: *SP* ∈*S P*
show $Q \lesssim[\![\cdot]\!]R{<}SRel,TRel{>}\ P$
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*)
next
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 *transS A2 A5* **have** (*SP*, *SR*) ∈ *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*) ∈ *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** (⟦*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*
with *symmT* **have** (*TQ*, ⟦*SP*⟧) ∈ *TRel*
unfolding *sym-def*
by *simp*
moreover assume *A7*: *TR* ∈*T R*
with *fullAbs A2 A5* **have** (⟦*SP*⟧, *TR*) ∈ *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*⟧, *TR*) ∈ *TRel*
using *trancl-id*[*of TRel*$^=$] *reflcl-of-refl-rel*[*of TRel*] *trancl-reflcl*[*of TRel*]
by *auto*
ultimately have (*TQ*, *TR*) ∈ *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* ∈*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 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
              ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
              ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
              ∧ ((refl SRel ∧ trans TRel)
                 ⟷ trans (Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q}))
proof −
  define Rel where Rel = (indRelSTEQ SRel TRel) − ({(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q}
               ∪ {(P, Q). ∃ S1 S2. S1 ∈S P ∧ S2 ∈S Q ∧ (S1, S2) ∉ SRel}
               ∪ {(P, Q). ∃ T1 T2. T1 ∈T P ∧ T2 ∈T Q ∧ (T1, T2) ∉ TRel})
  from Rel-def have ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
    by (simp add: indRelSTEQ.encR[where SRel=SRel and TRel=TRel])
  moreover from Rel-def have SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
  proof auto
    fix S1 S2
    assume (S1, S2) ∈ SRel
    thus SourceTerm S1 ∼⟦·⟧<SRel,TRel> 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) ∈ 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])
  qed
  moreover
  have (refl SRel ∧ trans TRel) ⟷ trans (Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q})
  proof (rule iffI, erule conjE)
    assume reflS: refl SRel and transT: trans TRel
    have Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q} = indRelSTEQ SRel TRel
    proof (auto simp add: Rel-def)
      fix S
      show TargetTerm (⟦S⟧) ∼⟦·⟧<SRel,TRel> SourceTerm S
        by (rule indRelSTEQ.encL)
    next
      fix S1 S2
      assume SourceTerm S1 ∼⟦·⟧<SRel,TRel> SourceTerm S2
      with fullAbs reflS transT have (S1, S2) ∈ 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) ∉ SRel
      ultimately show False
        by simp
    next
      fix T1 T2
      assume TargetTerm T1 ∼⟦·⟧<SRel,TRel> TargetTerm T2
      with fullAbs reflS transT have (T1, T2) ∈ 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) ∉ TRel
      ultimately show False
        by simp
    qed
    thus trans (Rel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q})
        using indRelSTEQ-trans[where SRel=SRel and TRel=TRel]
        unfolding trans-def
      by blast
  next
```

assume *transR*: *trans* (*Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*})
show *refl SRel* ∧ *trans TRel*
unfolding *trans-def refl-on-def*
proof *auto*
fix *S*
from *Rel-def* **have** (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*}
by (*simp add*: *indRelSTEQ.encR*)
moreover have (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*}
by *simp*
ultimately have (*SourceTerm S*, *SourceTerm S*) ∈ *Rel*
using *transR*
unfolding *trans-def*
by *blast*
with *Rel-def* **show** (*S*, *S*) ∈ *SRel*
by *simp*
next
fix *TP TQ TR*
assume (*TP*, *TQ*) ∈ *TRel*
with *Rel-def* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*}
by (*simp add*: *indRelSTEQ.target*)
moreover assume (*TQ*, *TR*) ∈ *TRel*
with *Rel-def* **have** (*TargetTerm TQ*, *TargetTerm TR*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*}
by (*simp add*: *indRelSTEQ.target*)
ultimately have (*TargetTerm TP*, *TargetTerm TR*) ∈ *Rel*
using *transR*
unfolding *trans-def*
by *blast*
with *Rel-def* **show** (*TP*, *TR*) ∈ *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* × *'procS*) *set*
and *TRel* :: (*'procT* × *'procT*) *set*
assumes *fullAbs*: *fully-abstract SRel TRel*
and *reflT*: *refl TRel*
and *transT*: *trans TRel*
shows ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ *trans* (*Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*})
proof −
define *Rel* **where** *Rel* = (*indRelSTEQ SRel TRel*) − {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T* *P* ∧ *S* ∈*S* *Q*}
from *fullAbs reflT* **have** *reflS*: *refl SRel*
unfolding *refl-on-def*
by *auto*
from *Rel-def* **have** ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
by (*simp add*: *indRelSTEQ.encR*[**where** *SRel*=*SRel* **and** *TRel*=*TRel*])
moreover from *Rel-def* **have** *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
proof *auto*
fix *S1 S2*
assume (*S1*, *S2*) ∈ *SRel*
thus *SourceTerm S1* ∼⟦·⟧<*SRel*,*TRel*> *SourceTerm S2*
by (*simp add*: *indRelSTEQ.source*[**where** *SRel*=*SRel* **and** *TRel*=*TRel*])
next
fix *S1 S2*
assume *SourceTerm S1* ∼⟦·⟧<*SRel*,*TRel*> *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 × 'procS) set
    and TRel :: ('procT × 'procT) set
    and Rel  :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes fullAbs: fully-abstract SRel TRel
      and rel:     Rel = ((indRelSTEQ SRel TRel)
                   − {(P, Q). (P ∈ ProcS ∧ Q ∈ ProcS) ∨ (P ∈ ProcT ∧ Q ∈ ProcT)})
                   ∪ {(P, Q). (∃ SP SQ. SP ∈S P ∧ SQ ∈S Q ∧ (SP, SQ) ∈ SRel)
                      ∨ (∃ TP TQ. TP ∈T P ∧ TQ ∈T Q ∧ (TP, TQ) ∈ TRel)}
  shows (refl SRel ∧ trans TRel) = trans Rel
    unfolding trans-def
proof auto
  fix P Q R
  assume A1: refl SRel and A2: ∀ x y. (x, y) ∈ TRel ⟶ (∀ z. (y, z) ∈ TRel ⟶ (x, z) ∈ TRel)
     and A3: (P, Q) ∈ Rel and A4: (Q, R) ∈ Rel
  from fullAbs rel have A5: ∀ SP SQ. (SourceTerm SP, SourceTerm SQ) ∈ Rel ⟶ (⟦SP⟧, ⟦SQ⟧) ∈ TRel
    by simp
  from rel have A6: ∀ TP TQ. (TargetTerm TP, TargetTerm TQ) ∈ Rel ⟶ (TP, TQ) ∈ TRel
    by simp
  have A7: ∀ SP TQ. (SourceTerm SP, TargetTerm TQ) ∈ Rel ⟶ (⟦SP⟧, TQ) ∈ TRel
  proof clarify
    fix SP TQ
    assume (SourceTerm SP, TargetTerm TQ) ∈ Rel
    with rel have SourceTerm SP ∼⟦·⟧<SRel,TRel> TargetTerm TQ
      by simp
    with A1 A2 fullAbs show (⟦SP⟧, TQ) ∈ 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: ∀ TP SQ. (TargetTerm TP, SourceTerm SQ) ∈ Rel ⟶ (TP, ⟦SQ⟧) ∈ TRel
  proof clarify
    fix TP SQ
    assume (TargetTerm TP, SourceTerm SQ) ∈ Rel
    with rel have TargetTerm TP ∼⟦·⟧<SRel,TRel> SourceTerm SQ
      by simp
    with A1 A2 fullAbs show (TP, ⟦SQ⟧) ∈ 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) ∈ Rel
  proof (cases P)
    case (SourceTerm SP)
    assume A9: SP ∈S P
    show (P, R) ∈ Rel
    proof (cases Q)
      case (SourceTerm SQ)
      assume A10: SQ ∈S Q
      with A3 A5 A9 have A11: (⟦SP⟧, ⟦SQ⟧) ∈ TRel
        by simp
      show (P, R) ∈ 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) ∈ 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) ∈ 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 (TP, ⟦SR⟧) ∈ TRel
      by blast
    with A9 have P ∼⟦·⟧<SRel,TRel> TargetTerm (⟦SR⟧)
      by (simp add: indRelSTEQ.target)
    moreover from A12 have TargetTerm (⟦SR⟧) ∼⟦·⟧<SRel,TRel> R
      by (simp add: indRelSTEQ.encL)
    ultimately have P ∼⟦·⟧<SRel,TRel> R
      by (rule indRelSTEQ.trans)
    with rel A9 A12 show (P, R) ∈ Rel
      by simp
  next
    case (TargetTerm TR)
    assume A12: TR ∈T R
    with A4 A7 A10 have (⟦SQ⟧, TR) ∈ TRel
      by simp
    with A2 A11 have (TP, TR) ∈ TRel
      by blast
    with rel A9 A12 show (P, R) ∈ 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) ∈ 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 (TP, ⟦SR⟧) ∈ TRel
      by blast
    with A9 have P ∼⟦·⟧<SRel,TRel> TargetTerm (⟦SR⟧)
      by (simp add: indRelSTEQ.target)
    moreover from A12 have TargetTerm (⟦SR⟧) ∼⟦·⟧<SRel,TRel> R
      by (simp add: indRelSTEQ.encL)
    ultimately have P ∼⟦·⟧<SRel,TRel> R
      by (rule indRelSTEQ.trans)
    with rel A9 A12 show (P, R) ∈ Rel
      by simp
  next
    case (TargetTerm TR)
    assume A12: TR ∈T R
    with A4 A6 A10 have (TQ, TR) ∈ TRel
      by simp
    with A2 A11 have (TP, TR) ∈ 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 $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$
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*: *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 add 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 $fully\text{-}abstract\ \{(S1,\ S2).\ (SourceTerm\ S1,\ SourceTerm\ S2) \in Rel\}$
$\{(T1,\ T2).\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel\}$
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 $fully\text{-}abstract\ SRel\ TRel =$
$(\exists Rel.\ (\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
$\wedge\ SRel = \{(S1,\ S2).\ (SourceTerm\ S1,\ SourceTerm\ S2) \in Rel\}$
$\wedge\ TRel = \{(T1,\ T2).\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel\}$
$\wedge\ trans\ (Rel \cup \{(P,\ Q).\ \exists S.\ \llbracket S \rrbracket \in T\ P \wedge S \in S\ Q\}))$

proof (*rule iffI*)
  assume *fully-abstract SRel TRel*
  with *preord* **show** $\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
    $\wedge$ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) $\in$ *Rel*}
    $\wedge$ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) $\in$ *Rel*}
    $\wedge$ *trans* (*Rel* $\cup$ {(*P*, *Q*). $\exists$ *S*. $[\![S]\!]$ $\in$*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* ($[\![S]\!]$)) $\in$ *Rel*)
    $\wedge$ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) $\in$ *Rel*}
    $\wedge$ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) $\in$ *Rel*}
    $\wedge$ *trans* (*Rel* $\cup$ {(*P*, *Q*). $\exists$ *S*. $[\![S]\!]$ $\in$*T P* $\wedge$ *S* $\in$*S Q*})
  from *this* **obtain** *Rel*
    where $\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *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*. $[\![S]\!]$ $\in$*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* ($[\![S]\!]$)) $\in$ *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*. $[\![S]\!]$ $\in$*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* ($[\![S]\!]$)) $\in$ *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* $\cup$ {(*P*, *Q*). $\exists$ *S*. $[\![S]\!]$ $\in$*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*. $[\![S]\!]$ $\in$*T P* $\wedge$ *S* $\in$*S Q*} $\cup$ {})
      unfolding *trans-def*
    by *blast*
  ultimately show $\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *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*. $[\![S]\!]$ $\in$*T P* $\wedge$ *S* $\in$*S Q*} $\cup$ *Rel'*))
      using *A1 A2 A3*
    by *blast*
next
  assume $\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)

$\wedge$ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
$\wedge$ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
$\wedge$ (∃ *Rel'*. (∀ (*P*, *Q*) ∈ *Rel'*. *P* ∈ *ProcS* ⟷ *Q* ∈ *ProcT*)
$\wedge$ *trans* (*Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*))
from *this* **obtain** *Rel Rel'*
where *B1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *B2*: *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
and *B3*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *B4*: ∀ (*P*, *Q*) ∈ *Rel'*. *P* ∈ *ProcS* ⟷ *Q* ∈ *ProcT*
and *B5*: *trans* (*Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*)
by *blast*
have *fully-abstract SRel TRel*
proof *auto*
fix *S1 S2*
have (*TargetTerm* (⟦*S1*⟧), *SourceTerm S1*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
moreover assume (*S1*, *S2*) ∈ *SRel*
with *B2* **have** (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
moreover from *B1*
have (*SourceTerm S2*, *TargetTerm* (⟦*S2*⟧)) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
ultimately have (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel* ∪ *Rel'*
using *B5*
unfolding *trans-def*
by *blast*
with *B3 B4* **show** (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
by *blast*
next
fix *S1 S2*
from *B1*
have (*SourceTerm S1*, *TargetTerm* (⟦*S1*⟧)) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
moreover assume (⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*
with *B3*
have (*TargetTerm* (⟦*S1*⟧), *TargetTerm* (⟦*S2*⟧)) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
moreover
have (*TargetTerm* (⟦*S2*⟧), *SourceTerm S2*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
ultimately have (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel* ∪ *Rel'*
using *B5*
unfolding *trans-def*
by *blast*
with *B2 B4* **show** (*S1*, *S2*) ∈ *SRel*
by *blast*
qed
moreover have *refl SRel*
unfolding *refl-on-def*
proof *auto*
fix *S*
from *B1* **have** (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
moreover
have (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} ∪ *Rel'*
by *simp*
ultimately have (*SourceTerm S*, *SourceTerm S*) ∈ *Rel* ∪ *Rel'*
using *B5*
unfolding *trans-def*
by *blast*
with *B2 B4* **show** (*S*, *S*) ∈ *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* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$
and *Cond* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool$
and *CondA* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool$
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* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$
and *Cond* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool$
and *CondA CondB* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool$
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* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$
and *Cond* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool$
and *CondA CondB* :: $(('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool$
assumes *A1*: $\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in RelA$
and *A2*: $\forall (P, Q) \in RelA. CondA (P, Q)$
and *A3*: $\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in RelB$
and *A4*: $\forall (P, Q) \in RelB. CondB (P, Q)$
shows $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in Rel) \wedge (\forall (P, Q) \in Rel. CondA (P, Q))$
$\wedge (\forall (P, Q) \in Rel. CondB (P, Q))$
and $Cond\ indRelR \Longrightarrow (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in Rel)$
$\wedge (\forall (P, Q) \in Rel. CondA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q)) \wedge Cond\ Rel)$
proof −
have *A5*: $\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in indRelR$
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 $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm (⟦S⟧)) \in Rel)$
$\wedge (\forall (P, Q) \in Rel. CondA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q))$
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=λR. ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ R] A2 A4
by blast
from A2 A4 A5 A6 A7
show Cond indRelR ⟹ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ (∀ (P, Q) ∈ Rel. CondA (P, Q)) ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q)) ∧ 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=λR. ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ R ∧ Cond R]
by blast
qed

lemma (**in** encoding) indRelR-cond-respects-predA-and-reflects-predB:
fixes PredA PredB :: ('procS, 'procT) Proc ⇒ bool
shows ((∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA)
∧ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-reflects-pred Rel PredB))
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA
∧ rel-reflects-pred Rel PredB)
proof (rule iffI, erule conjE)
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA
from this **obtain** RelA **where** A1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ RelA
and A2: rel-respects-pred RelA PredA
by blast
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-reflects-pred Rel PredB
from this **obtain** RelB **where** A3: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ RelB
and A4: rel-reflects-pred RelB PredB
by blast
from A2 **have** ∀ (P, Q) ∈ RelA. PredA P ⟷ PredA Q
by blast
moreover from A4 **have** ∀ (P, Q) ∈ RelB. PredB Q ⟶ PredB P
by blast
ultimately have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ (∀ (P, Q)∈Rel. PredA P = PredA Q) ∧ (∀ (P, Q)∈Rel. PredB Q ⟶ PredB P)
using combine-conditions-on-relations-indRelR(1)[**where** RelA=RelA **and** RelB=RelB **and**
CondA=λ(P, Q). PredA P ⟷ PredA Q **and** CondB=λ(P, Q). PredB Q ⟶ PredB P] A1 A3
by simp
thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA
∧ rel-reflects-pred Rel PredB
by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA
∧ rel-reflects-pred Rel PredB
thus (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ rel-respects-pred Rel PredA) ∧
(∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ 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} ∧ enc-reflects-divergence)
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
∧ rel-reflects-divergence Rel (STCal Source Target))
proof −

have $\forall$ *Rel. rel-reflects-divergence Rel* (*STCal Source Target*)
= *rel-reflects-pred Rel divergentST*
by (*simp add*: *divergentST-STCal-divergent*)
moreover have $\forall$ *Rel.* (*rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
= *rel-respects-pred Rel* ($\lambda P.\ P\Downarrow$*.success*))
by (*simp add*: *STCalWB-reachesBarbST*)
ultimately show (*enc-weakly-respects-barb-set* {*success*} $\wedge$ *enc-reflects-divergence*)
= ($\exists$ *Rel.* ($\forall$ *S.* (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
$\wedge$ *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=$\lambda P.\ P\Downarrow$*.success* **and** *PredB*=*divergentST*]
by *simp*
qed

lemma (**in** *encoding-wrt-barbs*) *SS-DR-iff-source-target-rel*:
fixes *success* :: *'barbs*
shows (*enc-respects-barb-set* {*success*} $\wedge$ *enc-reflects-divergence*)
= ($\exists$ *Rel.* ($\forall$ *S.* (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
$\wedge$ *rel-reflects-divergence Rel* (*STCal Source Target*))
proof −
have $\forall$ *Rel. rel-reflects-divergence Rel* (*STCal Source Target*)
= *rel-reflects-pred Rel divergentST*
by (*simp add*: *divergentST-STCal-divergent*)
moreover have $\forall$ *Rel.* (*rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
= *rel-respects-pred Rel* ($\lambda P.\ P\downarrow$*.success*))
by (*simp add*: *STCalWB-hasBarbST*)
ultimately show (*enc-respects-barb-set* {*success*} $\wedge$ *enc-reflects-divergence*)
= ($\exists$ *Rel.* ($\forall$ *S.* (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *Rel*)
$\wedge$ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
$\wedge$ *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=$\lambda P.\ P\downarrow$*.success* **and** *PredB*=*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 sensitiveness, 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 sensitiveness.

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* $\lesssim[\![\cdot]\!]$*RT*<*TRel*> *Q* **and** *P* $\longmapsto$(*Calculus* (*STCalWB SWB TWB*))$*$ *P'*
and *P'*$\downarrow$<*STCalWB SWB TWB*>*success*
thus *Q*$\Downarrow$<*STCalWB SWB TWB*>*success*
proof (*induct arbitrary*: *P'*)

case (*encR S*)
assume $SourceTerm\ S \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ P'$ **and** $P'\downarrow<STCalWB\ SWB\ TWB>success$
hence $S\Downarrow<SWB>success$
using *STCalWB-reachesBarbST*
by *blast*
with *encRS* **have** $[\![S]\!]\Downarrow<TWB>success$
by *simp*
thus $TargetTerm\ ([\![S]\!])\Downarrow<STCalWB\ SWB\ TWB>success$
using *STCalWB-reachesBarbST*
by *blast*
next
case (*source S*)
assume $SourceTerm\ S \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ P'$ **and** $P'\downarrow<STCalWB\ SWB\ TWB>success$
thus $SourceTerm\ S\Downarrow<STCalWB\ SWB\ TWB>success$
by *blast*
next
case (*target T1 T2*)
assume $(T1,\ T2) \in TRel$
moreover assume $TargetTerm\ T1 \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ P'$
and $P'\downarrow<STCalWB\ SWB\ TWB>success$
hence $T1\Downarrow<TWB>success$
using *STCalWB-reachesBarbST*
by *blast*
ultimately have $T2\Downarrow<TWB>success$
using *trelPS*
by *simp*
thus $TargetTerm\ T2\Downarrow<STCalWB\ SWB\ TWB>success$
using *STCalWB-reachesBarbST*
by *blast*
next
case (*trans P Q R*)
assume $P \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ P'$ **and** $P'\downarrow<STCalWB\ SWB\ TWB>success$
and $\bigwedge P'.\ P \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ P' \Longrightarrow P'\downarrow<STCalWB\ SWB\ TWB>success$
$\Longrightarrow Q\Downarrow<STCalWB\ SWB\ TWB>success$
hence $Q\Downarrow<STCalWB\ SWB\ TWB>success$
by *simp*
moreover assume $\bigwedge Q'.\ Q \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ Q' \Longrightarrow Q'\downarrow<STCalWB\ SWB\ TWB>success$
$\Longrightarrow R\Downarrow<STCalWB\ SWB\ TWB>success$
ultimately show $R\Downarrow<STCalWB\ SWB\ TWB>success$
by *blast*
qed
next
fix $P\ Q\ Q'$
assume $P \lesssim[\![\cdot]\!]RT<TRel>\ Q$ **and** $Q \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ Q'$
and $Q'\downarrow<STCalWB\ SWB\ TWB>success$
thus $P\Downarrow<STCalWB\ SWB\ TWB>success$
proof (*induct arbitrary*: Q')
case (*encR S*)
assume $TargetTerm\ ([\![S]\!]) \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ Q'$
and $Q'\downarrow<STCalWB\ SWB\ TWB>success$
hence $[\![S]\!]\Downarrow<TWB>success$
using *STCalWB-reachesBarbST*
by *blast*
with *encRS* **have** $S\Downarrow<SWB>success$
by *simp*
thus $SourceTerm\ S\Downarrow<STCalWB\ SWB\ TWB>success$
using *STCalWB-reachesBarbST*
by *blast*
next
case (*source S*)
assume $SourceTerm\ S \longmapsto(Calculus\ (STCalWB\ SWB\ TWB))*\ Q'$ **and** $Q'\downarrow<STCalWB\ SWB\ TWB>success$

thus *SourceTerm S*⇓<*STCalWB SWB TWB*>*success*
by *blast*
next
case (*target T1 T2*)
assume (*T1*, *T2*) ∈ *TRel*
moreover assume *TargetTerm T2* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *Q′*
and *Q′*↓<*STCalWB SWB TWB*>*success*
hence *T2*⇓<*TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *T1*⇓<*TWB*>*success*
using *trelRS*
by *blast*
thus *TargetTerm T1*⇓<*STCalWB SWB TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
next
case (*trans P Q R R′*)
assume *R* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *R′* **and** *R′*↓<*STCalWB SWB TWB*>*success*
and ⋀*R′*. *R* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *R′* ⟹ *R′*↓<*STCalWB SWB TWB*>*success*
⟹ *Q*⇓<*STCalWB SWB TWB*>*success*
hence *Q*⇓<*STCalWB SWB TWB*>*success*
by *simp*
moreover assume ⋀*Q′*. *Q* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *Q′* ⟹ *Q′*↓<*STCalWB SWB TWB*>*success*
⟹ *P*⇓<*STCalWB SWB TWB*>*success*
ultimately show *P*⇓<*STCalWB SWB TWB*>*success*
by *blast*
qed
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* ≲⟦·⟧*RT*<*TRel*> *Q* **and** *P* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *P′*
and *P′*↓<*STCalWB SWB TWB*>*x*
thus *Q*⇓<*STCalWB SWB TWB*>*x*
proof (*induct arbitrary*: *P′*)
case (*encR S*)
assume *SourceTerm S* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *P′* **and** *P′*↓<*STCalWB SWB TWB*>*x*
hence *S*⇓<*SWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
with *encRS* **have** ⟦*S*⟧⇓<*TWB*>*x*
by *simp*
thus *TargetTerm* (⟦*S*⟧)⇓<*STCalWB SWB TWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
next
case (*source S*)
assume *SourceTerm S* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *P′* **and** *P′*↓<*STCalWB SWB TWB*>*x*
thus *SourceTerm S*⇓<*STCalWB SWB TWB*>*x*
by *blast*
next
case (*target T1 T2*)
assume (*T1*, *T2*) ∈ *TRel*
moreover assume *TargetTerm T1* ⟼(*Calculus* (*STCalWB SWB TWB*))∗ *P′*

and P'↓<STCalWB SWB TWB>x
hence T1⇓<TWB>x
using STCalWB-reachesBarbST
by blast
ultimately have T2⇓<TWB>x
using trelPS
by simp
thus TargetTerm T2⇓<STCalWB SWB TWB>x
using STCalWB-reachesBarbST
by blast
next
case (trans P Q R)
assume P ⟼(Calculus (STCalWB SWB TWB))∗ P′ **and** P′↓<STCalWB SWB TWB>x
and ⋀P′. P ⟼(Calculus (STCalWB SWB TWB))∗ P′ ⟹ P′↓<STCalWB SWB TWB>x
⟹ Q⇓<STCalWB SWB TWB>x
hence Q⇓<STCalWB SWB TWB>x
by simp
moreover assume ⋀Q′. Q ⟼(Calculus (STCalWB SWB TWB))∗ Q′ ⟹ Q′↓<STCalWB SWB TWB>x
⟹ R⇓<STCalWB SWB TWB>x
ultimately show R⇓<STCalWB SWB TWB>x
by blast
qed
next
fix P Q x Q′
assume P ≲⟦·⟧RT<TRel> Q **and** Q ⟼(Calculus (STCalWB SWB TWB))∗ Q′
and Q′↓<STCalWB SWB TWB>x
thus P⇓<STCalWB SWB TWB>x
proof (induct arbitrary: Q′)
case (encR S)
assume TargetTerm (⟦S⟧) ⟼(Calculus (STCalWB SWB TWB))∗ Q′
and Q′↓<STCalWB SWB TWB>x
hence ⟦S⟧⇓<TWB>x
using STCalWB-reachesBarbST
by blast
with encRS **have** S⇓<SWB>x
by simp
thus SourceTerm S⇓<STCalWB SWB TWB>x
using STCalWB-reachesBarbST
by blast
next
case (source S)
assume SourceTerm S ⟼(Calculus (STCalWB SWB TWB))∗ Q′ **and** Q′↓<STCalWB SWB TWB>x
thus SourceTerm S⇓<STCalWB SWB TWB>x
by blast
next
case (target T1 T2)
assume (T1, T2) ∈ TRel
moreover assume TargetTerm T2 ⟼(Calculus (STCalWB SWB TWB))∗ Q′
and Q′↓<STCalWB SWB TWB>x
hence T2⇓<TWB>x
using STCalWB-reachesBarbST
by blast
ultimately have T1⇓<TWB>x
using trelRS
by blast
thus TargetTerm T1⇓<STCalWB SWB TWB>x
using STCalWB-reachesBarbST
by blast
next
case (trans P Q R R′)
assume R ⟼(Calculus (STCalWB SWB TWB))∗ R′ **and** R′↓<STCalWB SWB TWB>x

and ⋀R′. R ⟼(Calculus (STCalWB SWB TWB))∗ R′ ⟹ R′↓<STCalWB SWB TWB>x
⟹ Q⇓<STCalWB SWB TWB>x
hence Q⇓<STCalWB SWB TWB>x
by *simp*
moreover assume ⋀Q′. Q ⟼(Calculus (STCalWB SWB TWB))∗ Q′ ⟹ Q′↓<STCalWB SWB TWB>x
⟹ P⇓<STCalWB SWB TWB>x
ultimately show P⇓<STCalWB SWB TWB>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 ≲⟦·⟧RT<TRel> Q **and** P↓<STCalWB SWB TWB>success
thus Q↓<STCalWB SWB TWB>success
proof *induct*
case (*encR S*)
assume SourceTerm S↓<STCalWB SWB TWB>success
hence S↓<SWB>success
using *STCalWB-hasBarbST*
by *blast*
with *encRS* **have** ⟦S⟧↓<TWB>success
by *simp*
thus TargetTerm (⟦S⟧)↓<STCalWB SWB TWB>success
using *STCalWB-hasBarbST*
by *blast*
next
case (*source S*)
assume SourceTerm S↓<STCalWB SWB TWB>success
thus SourceTerm S↓<STCalWB SWB TWB>success
by *simp*
next
case (*target T1 T2*)
assume (T1, T2) ∈ TRel
moreover assume TargetTerm T1↓<STCalWB SWB TWB>success
hence T1↓<TWB>success
using *STCalWB-hasBarbST*
by *blast*
ultimately have T2↓<TWB>success
using *trelPS*
by *simp*
thus TargetTerm T2↓<STCalWB SWB TWB>success
using *STCalWB-hasBarbST*
by *blast*
next
case (*trans P Q R*)
assume P↓<STCalWB SWB TWB>success
and P↓<STCalWB SWB TWB>success ⟹ Q↓<STCalWB SWB TWB>success
and Q↓<STCalWB SWB TWB>success ⟹ R↓<STCalWB SWB TWB>success
thus R↓<STCalWB SWB TWB>success
by *simp*
qed
next
fix P Q

assume *P* ≲⟦·⟧*RT*<*TRel*> *Q* **and** *Q*↓<*STCalWB SWB TWB*>*success*
thus *P*↓<*STCalWB SWB TWB*>*success*
proof *induct*
case (*encR S*)
assume *TargetTerm* (⟦*S*⟧)↓<*STCalWB SWB TWB*>*success*
hence ⟦*S*⟧↓<*TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
with *encRS* **have** *S*↓<*SWB*>*success*
by *simp*
thus *SourceTerm S*↓<*STCalWB SWB TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
next
case (*source S*)
assume *SourceTerm S*↓<*STCalWB SWB TWB*>*success*
thus *SourceTerm S*↓<*STCalWB SWB TWB*>*success*
by *simp*
next
case (*target T1 T2*)
assume (*T1*, *T2*) ∈ *TRel*
moreover assume *TargetTerm T2*↓<*STCalWB SWB TWB*>*success*
hence *T2*↓<*TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
ultimately have *T1*↓<*TWB*>*success*
using *trelRS*
by *blast*
thus *TargetTerm T1*↓<*STCalWB SWB TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
next
case (*trans P Q R*)
assume *R*↓<*STCalWB SWB TWB*>*success*
and *R*↓<*STCalWB SWB TWB*>*success* ⟹ *Q*↓<*STCalWB SWB TWB*>*success*
and *Q*↓<*STCalWB SWB TWB*>*success* ⟹ *P*↓<*STCalWB SWB TWB*>*success*
thus *P*↓<*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 *trelPS*: *rel-preserves-barbs TRel TWB*
and *trelRS*: *rel-reflects-barbs TRel TWB*
shows *rel-respects-barbs* (*indRelRTPO TRel*) (*STCalWB SWB TWB*)
proof *auto*
fix *P Q x*
assume *P* ≲⟦·⟧*RT*<*TRel*> *Q* **and** *P*↓<*STCalWB SWB TWB*>*x*
thus *Q*↓<*STCalWB SWB TWB*>*x*
proof *induct*
case (*encR S*)
assume *SourceTerm S*↓<*STCalWB SWB TWB*>*x*
hence *S*↓<*SWB*>*x*
using *STCalWB-hasBarbST*
by *blast*
with *encRS* **have** ⟦*S*⟧↓<*TWB*>*x*
by *simp*
thus *TargetTerm* (⟦*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 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*↓<*STCalWB SWB TWB*>*x*
  and *R*↓<*STCalWB SWB TWB*>*x* ⟹ *Q*↓<*STCalWB SWB TWB*>*x*
  and *Q*↓<*STCalWB SWB TWB*>*x* ⟹ *P*↓<*STCalWB SWB TWB*>*x*
thus *P*↓<*STCalWB SWB TWB*>*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 bisimultion 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* (⟦*S*⟧)) ∈ *Rel*)
   ∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
   ∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$)
   ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *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* (⟦*S*⟧)) ∈ *Rel*
  by (*simp add*: *indRelRTPO.encR*)
 from *Rel-def* **have** *B2*: ∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*
  by (*simp add*: *indRelRTPO.target*)
 from *Rel-def* **have** *B3*: ∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*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* ⟶ (⟦*S*⟧, *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* **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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
   ∧ (∀ *T1 T2*. (*T1*, *T2*) ∈ *TRel* ⟶ (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*)
   ∧ (∀ *T1 T2*. (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel* ⟶ (*T1*, *T2*) ∈ $TRel^+$)
   ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ $TRel^*$)
   ∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
   ∧ *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 (⟦S⟧)) ∈ Rel)
         ∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
         ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
         ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
         ∧ weak-reduction-bisimulation Rel (STCal Source Target)
         ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  from this obtain Rel where C1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
   and C2: ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
   and C3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
   and C4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*
   and C5: weak-reduction-bisimulation Rel (STCal Source Target)
   and C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
    by auto
  from C1 C2 C3 C4 C5 have ∃ Rel.(∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
   ∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
   ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
   ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
   ∧ weak-reduction-bisimulation Rel (STCal Source Target)
    by blast
  hence operational-corresponding (TRel*)
       ∧ weak-reduction-bisimulation (TRel⁺) Target
      using OC-iff-weak-reduction-bisimulation[where TRel=TRel]
    by auto
  moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                ∧ 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 auto
  moreover have rel-weakly-respects-barb-set TRel TWB {success}
  proof auto
    fix TP TQ TP'
    assume (TP, TQ) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TP ⟼(Calculus TWB)* TP' and TP'↓<TWB>success
    hence TargetTerm TP⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
      by blast
    ultimately have TargetTerm TQ⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TQ⇓<TWB>success
        using STCalWB-reachesBarbST
      by blast
  next
    fix TP TQ TQ'
    assume (TP, TQ) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TQ ⟼(Calculus TWB)* TQ' and TQ'↓<TWB>success
    hence TargetTerm TQ⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
      by blast
    ultimately have TargetTerm TP⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TP⇓<TWB>success
        using STCalWB-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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
          ∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
          ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
          ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
          ∧ weak-reduction-bisimulation Rel (STCal Source Target)
          ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
          ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  from this obtain Rel where C: (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
          ∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
          ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
          ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
          ∧ weak-reduction-bisimulation Rel (STCal Source Target)
          ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
          ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
    by auto
  hence C1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
    by simp
  from C have C2: ∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel
    by simp
  from C have C3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺
    by simp
  from C have C4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*
    by simp
  from C have C5: weak-reduction-bisimulation Rel (STCal Source Target)
    by simp
  from C have C7: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
    apply (rule conjE) apply (erule conjE)+ by blast
  from C have C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
    apply (rule conjE) apply (erule conjE)+ by blast
  from C1 C2 C3 C4 C5 have ∃ Rel.(∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
   ∧ (∀ T1 T2. (T1, T2) ∈ TRel ⟶ (TargetTerm T1, TargetTerm T2) ∈ Rel)
   ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel ⟶ (T1, T2) ∈ TRel⁺)
   ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel*)
   ∧ weak-reduction-bisimulation Rel (STCal Source Target)
    by blast
  hence operational-corresponding (TRel*)
        ∧ weak-reduction-bisimulation (TRel⁺) Target
      using OC-iff-weak-reduction-bisimulation[where TRel=TRel]
    by auto
  moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                ∧ 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 auto
  moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
                ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
    apply (rule exI) using C1 C7 by blast
  hence enc-weakly-respects-barbs
      using enc-weakly-respects-barbs-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) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TP ⟼(Calculus TWB)∗ TP′ and TP′↓<TWB>success
    hence TargetTerm TP⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
```

by *blast*
ultimately have *TargetTerm TQ⇓<STCalWB SWB TWB>success*
using *C6*
by *blast*
thus *TQ⇓<TWB>success*
using *STCalWB-reachesBarbST*
by *blast*
next
fix *TP TQ TQ'*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ* ⟼(*Calculus TWB*)∗ *TQ'* **and** *TQ'↓<TWB>success*
hence *TargetTerm TQ⇓<STCalWB SWB TWB>success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP⇓<STCalWB SWB TWB>success*
using *C6*
by *blast*
thus *TP⇓<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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP'* **and** *TP'↓<TWB>x*
hence *TargetTerm TP⇓<STCalWB SWB TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TQ⇓<STCalWB SWB TWB>x*
using *C7*
by *blast*
thus *TQ⇓<TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
next
fix *TP TQ x TQ'*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ* ⟼(*Calculus TWB*)∗ *TQ'* **and** *TQ'↓<TWB>x*
hence *TargetTerm TQ⇓<STCalWB SWB TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP⇓<STCalWB SWB TWB>x*
using *C7*
by *blast*
thus *TP⇓<TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
qed
ultimately show *operational-corresponding* ($TRel^*$)
∧ *weak-reduction-bisimulation* ($TRel^+$) *Target*
∧ *enc-weakly-respects-barbs* ∧ *enc-weakly-respects-barb-set* {*success*}
∧ *rel-weakly-respects-barbs TRel TWB* ∧ *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* (⟦*S*⟧)) ∈ *Rel*)
       ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
       ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *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*)+)
  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*] *preorder-on-def*
   by *blast*
  from *A5* **have** *A7*: $TRel^{*}$ = *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* (⟦*S*⟧)) ∈ *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* ⟶ (⟦*S*⟧, *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* (⟦*S*⟧)) ∈ *Rel*)
     ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
     ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *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* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
      ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *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*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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*}
by *auto*
from *C1 C2 C3 C4 C5* **have** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *operational-corresponding TRel* ∧ *preorder TRel* ∧ *weak-reduction-bisimulation TRel Target*
using *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP′* **and** *TP′*↓<*TWB*>*success*
hence *TargetTerm TP*⇓<*STCalWB SWB TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TQ*⇓<*TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
next
fix *TP TQ TQ′*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ* ⟼(*Calculus TWB*)∗ *TQ′* **and** *TQ′*↓<*TWB*>*success*
hence *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP*⇓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TP*⇓<*TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
qed
ultimately show *operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-bisimulation TRel Target*
∧ *enc-weakly-respects-barb-set* {*success*} ∧ *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* × ′*procT*) *set*

```
  shows (operational-corresponding TRel ∧ preorder TRel ∧ weak-reduction-bisimulation TRel Target
        ∧ enc-weakly-respects-barbs ∧ rel-weakly-respects-barbs TRel TWB
        ∧ enc-weakly-respects-barb-set {success}
        ∧ rel-weakly-respects-barb-set TRel TWB {success})
       = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
         ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
         ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
         ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
         ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
         ∧ 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: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ 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 ⟶ (⟦S⟧, 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-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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
      ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
      ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
      ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
      ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
      ∧ 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
        ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
        ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
        ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
        ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
        ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
```

from *this* **obtain** *Rel* **where** *C1*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
and *C4*: *weak-reduction-bisimulation Rel* (*STCal Source Target*) **and** *C5*: *preorder Rel*
and *C6*: *rel-weakly-respects-barbs Rel* (*STCalWB SWB TWB*)
by *auto*
from *C1 C2 C3 C4 C5* **have** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *operational-corresponding TRel* ∧ *preorder TRel* ∧ *weak-reduction-bisimulation TRel Target*
using *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP′* **and** *TP′*↓<*TWB*>*x*
hence *TargetTerm TP*⇓<*STCalWB SWB TWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*x*
using *C6*
by *blast*
thus *TQ*⇓<*TWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
next
fix *TP TQ x TQ′*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ* ⟼(*Calculus TWB*)∗ *TQ′* **and** *TQ′*↓<*TWB*>*x*
hence *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP*⇓<*STCalWB SWB TWB*>*x*
using *C6*
by *blast*
thus *TP*⇓<*TWB*>*x*
using *STCalWB-reachesBarbST*
by *blast*
qed
moreover hence *rel-weakly-respects-barb-set TRel TWB* {*success*}
by *blast*
ultimately show *operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-bisimulation TRel Target*
∧ *enc-weakly-respects-barbs* ∧ *rel-weakly-respects-barbs TRel TWB*
∧ *enc-weakly-respects-barb-set* {*success*} ∧ *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 simultion 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* ∧ *preorder TRel*
      ∧ *weak-reduction-correspondence-simulation TRel Target*
      ∧ *enc-weakly-respects-barb-set* {*success*}
      ∧ *rel-weakly-respects-barb-set TRel TWB* {*success*})
    = (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
      ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
      ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
      ∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
      ∧ *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*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *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* ⟶ (⟦*S*⟧, *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** *weakly-operational-sound TRel*
    and *weak-reduction-simulation TRel Target*
    and ∀ *P Q Q'*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ⟼*Target*∗ *Q'*
      ⟶ (∃ *P'' Q''*. *P* ⟼*Target*∗ *P''* ∧ *Q'* ⟼*Target*∗ *Q''* ∧ (*P''*, *Q''*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
    ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
    ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)

∧ *weak-reduction-correspondence-simulation 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* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
from *this* **obtain** *Rel* **where** *C1*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*)
by *blast*
hence *weakly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-correspondence-simulation TRel Target*
using *WOC-wrt-preorder-iff-reduction-correspondence-simulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP′* **and** *TP′*↓<*TWB*>*success*
hence *TargetTerm TP*⇓<*STCalWB SWB TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TQ*⇓<*TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
next
fix *TP TQ TQ′*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ* ⟼(*Calculus TWB*)∗ *TQ′* **and** *TQ′*↓<*TWB*>*success*
hence *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP*⇓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TP*⇓<*TWB*>*success*
using *STCalWB-reachesBarbST*
by *blast*

qed
ultimately show *weakly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-correspondence-simulation TRel Target*
∧ *enc-weakly-respects-barb-set* {*success*} ∧ *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* × *'procT*) *set*
shows (*weakly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-correspondence-simulation TRel Target*
∧ *enc-weakly-respects-barbs* ∧ *enc-weakly-respects-barb-set* {*success*}
∧ *rel-weakly-respects-barbs TRel TWB* ∧ *rel-weakly-respects-barb-set TRel TWB* {*success*})
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-weakly-respects-barbs Rel* (*STCalWB SWB TWB*)
∧ *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*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *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* ⟶ (⟦*S*⟧, *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** *weakly-operational-sound TRel*
and *weak-reduction-simulation TRel Target*
and ∀ *P Q Q'*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ⟼*Target*∗ *Q'*
⟶ (∃ *P'' Q''*. *P* ⟼*Target*∗ *P''* ∧ *Q'* ⟼*Target*∗ *Q''* ∧ (*P''*, *Q''*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
  ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
  ∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
  ∧ *rel-weakly-respects-barbs Rel* (*STCalWB SWB TWB*)
  ∧ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
  ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
  ∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
  ∧ *rel-weakly-respects-barbs Rel* (*STCalWB SWB TWB*)
  ∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
from *this* **obtain** *Rel* **where** *C1*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
 and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
 and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
 ∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
 ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
 ∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*)
 by *blast*
hence *weakly-operational-corresponding TRel* ∧ *preorder TRel*
  ∧ *weak-reduction-correspondence-simulation TRel Target*
  using *WOC-wrt-preorder-iff-reduction-correspondence-simulation*[**where** *TRel*=*TRel*]
 by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *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*) ∈ *TRel*
 with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
  by *simp*
 moreover assume *TP* ⟼(*Calculus TWB*)∗ *TP′* **and** *TP′*↓<*TWB*>*x*
 hence *TargetTerm TP*⇓<*STCalWB SWB TWB*>*x*
   using *STCalWB-reachesBarbST*
  by *blast*
 ultimately have *TargetTerm TQ*⇓<*STCalWB SWB TWB*>*x*
   using *C7*
  by *blast*
 thus *TQ*⇓<*TWB*>*x*
   using *STCalWB-reachesBarbST*
  by *blast*
next
 fix *TP TQ x TQ′*
 assume (*TP*, *TQ*) ∈ *TRel*
 with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
  by *simp*

moreover assume *TQ ⟼(Calculus TWB)∗ TQ′* **and** *TQ′↓<TWB>x*
hence *TargetTerm TQ⇓<STCalWB SWB TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
ultimately have *TargetTerm TP⇓<STCalWB SWB TWB>x*
using *C7*
by *blast*
thus *TP⇓<TWB>x*
using *STCalWB-reachesBarbST*
by *blast*
qed
moreover from *D2* **have** *rel-weakly-respects-barb-set TRel TWB {success}*
by *blast*
ultimately show *weakly-operational-corresponding TRel ∧ preorder TRel*
∧ weak-reduction-correspondence-simulation TRel Target
∧ enc-weakly-respects-barbs ∧ enc-weakly-respects-barb-set {success}
∧ rel-weakly-respects-barbs TRel TWB ∧ rel-weakly-respects-barb-set TRel 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 bisimultion 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 × ′procT) set*
shows *(strongly-operational-corresponding TRel ∧ preorder TRel*
∧ strong-reduction-bisimulation TRel Target
∧ enc-respects-barb-set {success} ∧ rel-respects-barb-set TRel TWB {success})
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success})
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*: *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*: *∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ 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 ⟶ (⟦S⟧, T) ∈ 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 ∀ *P Q Q'*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ⟼*Target Q'* ⟶ (∃ *P'*. *P* ⟼*Target P'* ∧ (*P'*, *Q'*) ∈ *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-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
  using *enc-and-TRel-impl-indRelRTPO-respects-success*[**where** *TRel*=*TRel* **and** *success*=*success*]
 by *blast*
show ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
  ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
  ∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
  ∧ *rel-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* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
  ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
  ∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
  ∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
from *this* **obtain** *Rel* **where** *C1*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
 and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
 and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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*}
 by *auto*
from *C1 C2 C3 C4 C5* **have** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
 ∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
 ∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
 ∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
 by *blast*
hence *strongly-operational-corresponding TRel* ∧ *preorder TRel*
  ∧ *strong-reduction-bisimulation TRel Target*
  using *SOC-wrt-preorder-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
 by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
  ∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
 apply (*rule exI*) **using** *C1 C6* **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*) ∈ *TRel*
 with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
  by *simp*
 moreover assume *TP*↓<*TWB*>*success*
 hence *TargetTerm TP*↓<*STCalWB SWB TWB*>*success*
   using *STCalWB-hasBarbST*
  by *blast*
 ultimately have *TargetTerm TQ*↓<*STCalWB SWB TWB*>*success*
   using *C6*
  by *blast*
 thus *TQ*↓<*TWB*>*success*
   using *STCalWB-hasBarbST*
  by *blast*
next

```
    fix TP TQ
    assume (TP, TQ) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TQ↓<TWB>success
    hence TargetTerm TQ↓<STCalWB SWB TWB>success
        using STCalWB-hasBarbST
      by blast
    ultimately have TargetTerm TP↓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TP↓<TWB>success
        using STCalWB-hasBarbST
      by blast
  qed
  ultimately show strongly-operational-corresponding TRel ∧ preorder TRel
   ∧ strong-reduction-bisimulation TRel Target
   ∧ enc-respects-barb-set {success} ∧ 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 × 'procT) set
  shows (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})
         = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
            ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
            ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
            ∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
            ∧ rel-respects-barbs Rel (STCalWB SWB TWB)
            ∧ 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: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ 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 ⟶ (⟦S⟧, T) ∈ 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 ∀ P Q Q'. (P, Q) ∈ TRel ∧ Q ⟼Target Q' ⟶ (∃ P'. P ⟼Target P' ∧ (P', Q') ∈ 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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-respects-barbs Rel* (*STCalWB SWB TWB*)
∧ *rel-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* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-respects-barbs Rel* (*STCalWB SWB TWB*)
∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
from *this* **obtain** *Rel* **where** *C1*: (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ (*TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*})
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *strongly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *strong-reduction-bisimulation TRel Target*
using *SOC-wrt-preorder-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP*↓<*TWB*>*x*
hence *TargetTerm TP*↓<*STCalWB SWB TWB*>*x*
using *STCalWB-hasBarbST*
by *blast*
ultimately have *TargetTerm TQ*↓<*STCalWB SWB TWB*>*x*
using *C6*
by *blast*
thus *TQ*↓<*TWB*>*x*

    using *STCalWB-hasBarbST*
   by *blast*
 next
  fix *TP TQ x*
  assume $(TP,\ TQ) \in TRel$
  with *C2* **have** $(TargetTerm\ TP,\ TargetTerm\ TQ) \in Rel$
   by *simp*
  moreover assume $TQ{\downarrow}{<}TWB{>}x$
  hence $TargetTerm\ TQ{\downarrow}{<}STCalWB\ SWB\ TWB{>}x$
    using *STCalWB-hasBarbST*
   by *blast*
  ultimately have $TargetTerm\ TP{\downarrow}{<}STCalWB\ SWB\ TWB{>}x$
    using *C6*
   by *blast*
  thus $TP{\downarrow}{<}TWB{>}x$
    using *STCalWB-hasBarbST*
   by *blast*
 qed
 moreover hence *rel-respects-barb-set TRel TWB* $\{success\}$
  by *blast*
 ultimately show *strongly-operational-corresponding TRel* $\wedge$ *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* $\{success\}$
  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* $::\ ('procT \times 'procT)\ set$
 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 $P \lesssim [\![\cdot]\!]RT{<}TRel{>}\ Q$ **and** $Q \longmapsto(STCal\ Source\ Target)\omega$
 thus $P \longmapsto(STCal\ Source\ Target)\omega$
 proof *induct*
  case (*encR S*)
  assume $TargetTerm\ ([\![S]\!]) \longmapsto(STCal\ Source\ Target)\omega$
  hence $[\![S]\!] \longmapsto(Target)\omega$
   by (*simp add*: *STCal-divergent*(*2*))
  with *encRD* **have** $S \longmapsto(Source)\omega$
   by *simp*
  thus $SourceTerm\ S \longmapsto(STCal\ Source\ Target)\omega$
   by (*simp add*: *STCal-divergent*(*1*))
 next
  case (*source S*)
  assume $SourceTerm\ S \longmapsto(STCal\ Source\ Target)\omega$
  thus $SourceTerm\ S \longmapsto(STCal\ Source\ Target)\omega$
   by *simp*
 next
  case (*target T1 T2*)
  assume $(T1,\ T2) \in TRel$
  moreover assume $TargetTerm\ T2 \longmapsto(STCal\ Source\ Target)\omega$
  hence $T2 \longmapsto(Target)\omega$
   by (*simp add*: *STCal-divergent*(*2*))
  ultimately have $T1 \longmapsto(Target)\omega$
    using *trelRD*
   by *blast*
  thus $TargetTerm\ T1 \longmapsto(STCal\ Source\ Target)\omega$

by (*simp add*: *STCal-divergent*(*2*))
next
case (*trans P Q R*)
assume $R \longmapsto(STCal\ Source\ Target)\omega$
and $R \longmapsto(STCal\ Source\ Target)\omega \Longrightarrow Q \longmapsto(STCal\ Source\ Target)\omega$
and $Q \longmapsto(STCal\ Source\ Target)\omega \Longrightarrow P \longmapsto(STCal\ Source\ Target)\omega$
thus $P \longmapsto(STCal\ Source\ Target)\omega$
by *simp*
qed
qed

lemma (**in** *encoding-wrt-barbs*) *OC-SS-DR-iff-source-target-rel*:
fixes *success* :: *'barbs*
and *TRel* :: (*'procT* × *'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* ($[\![S]\!]$)) $\in$ *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-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* ($[\![S]\!]$)) $\in$ *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* $\longrightarrow$ ($[\![S]\!]$, *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* **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* ($[\![S]\!]$)) $\in$ *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-bisimulation Rel* (*STCal Source Target*)
$\wedge$ *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* ($[\![S]\!]$)) $\in$ *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-bisimulation Rel* (*STCal Source Target*)
$\wedge$ *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* ($[\![S]\!]$)) $\in$ *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* $\longrightarrow$ ($[\![S]\!]$, *T*) $\in$ *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** $\exists$ *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-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *operational-corresponding* (*TRel**)
$\wedge$ *weak-reduction-bisimulation* (*TRel*$^+$) *Target*
using *OC-iff-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *auto*
moreover have $\exists$ *Rel*. ($\forall$ *S*. (*SourceTerm S*, *TargetTerm* ($[\![S]\!]$)) $\in$ *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 *auto*
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* $\longmapsto$(*Calculus TWB*)$*$ *TP'* **and** *TP'*$\downarrow$<*TWB*>*success*
hence *TargetTerm TP*$\Downarrow$<*STCalWB SWB TWB*>*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* $\longmapsto$(*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*⇓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TP*⇓<*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**)
∧ *weak-reduction-bisimulation* (*TRel*$^+$) *Target*
∧ *enc-weakly-respects-barb-set* {*success*} ∧ *rel-weakly-respects-barb-set TRel TWB* {*success*}
∧ *enc-reflects-divergence* ∧ *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* × *'procT*) *set*
shows (*weakly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *weak-reduction-correspondence-simulation TRel Target*
∧ *enc-weakly-respects-barb-set* {*success*}
∧ *rel-weakly-respects-barb-set TRel TWB* {*success*}
∧ *enc-reflects-divergence* ∧ *rel-reflects-divergence TRel Target*)
= (∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *weak-reduction-correspondence-simulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *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*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
by (*simp add*: *indRelRTPO.encR*)
from *Rel-def A8* **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 A9* **have** *B3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *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** *weakly-operational-sound TRel* **and** *preorder TRel*
and *weak-reduction-simulation TRel Target*
and ∀ *P Q Q'*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ⟼*Target*∗ *Q'*

```
        ⟶ (∃ P'' Q''. P ⟼Target∗ P'' ∧ Q' ⟼Target∗ Q'' ∧ (P'', Q'') ∈ 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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
      ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
      ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
      ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target) ∧ preorder Rel
      ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
      ∧ rel-reflects-divergence Rel (STCal Source Target)
   apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
        ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
        ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
        ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target) ∧ preorder Rel
        ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
        ∧ rel-reflects-divergence Rel (STCal Source Target)
  from this obtain Rel where C1: ∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel
   and C2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
   and C3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ 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 ∃ Rel.(∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
   ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
   ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel) ∧ preorder Rel
   ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)
   by blast
  hence weakly-operational-corresponding TRel ∧ preorder TRel
       ∧ weak-reduction-correspondence-simulation TRel Target
    using WOC-wrt-preorder-iff-reduction-correspondence-simulation[where TRel=TRel]
   by simp
  moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
               ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
               ∧ rel-reflects-divergence Rel (STCal Source Target)
   apply (rule exI) using C1 C6 C7 by blast
  hence enc-weakly-respects-barb-set {success} ∧ 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) ∈ TRel
   with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
    by simp
   moreover assume TP ⟼(Calculus TWB)∗ TP' and TP'↓<TWB>success
   hence TargetTerm TP⇓<STCalWB SWB TWB>success
     using STCalWB-reachesBarbST
```

```
      by blast
    ultimately have TargetTerm TQ⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TQ⇓<TWB>success
        using STCalWB-reachesBarbST
      by blast
  next
    fix TP TQ TQ′
    assume (TP, TQ) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TQ ⟼(Calculus TWB)∗ TQ′ and TQ′↓<TWB>success
    hence TargetTerm TQ⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
      by blast
    ultimately have TargetTerm TP⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TP⇓<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 ∧ preorder TRel
   ∧ weak-reduction-correspondence-simulation TRel Target
   ∧ enc-weakly-respects-barb-set {success} ∧ rel-weakly-respects-barb-set TRel TWB {success}
   ∧ enc-reflects-divergence ∧ 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 × ′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}
         ∧ enc-reflects-divergence ∧ rel-reflects-divergence TRel Target)
        = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
           ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
           ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
           ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
           ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
           ∧ 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*: ∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*
by (*simp add*: *indRelRTPO.encR*)
from *Rel-def A8* **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 A9* **have** *B3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ ([[*S*]], *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** *preorder 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 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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ ([[*S*]], *T*) ∈ *TRel*)
∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
apply (*rule exI*) **using** *B1 B2 B3 B4 B5 B6 B7* **by** *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ ([[*S*]], *T*) ∈ *TRel*)
∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *C1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ ([[*S*]], *T*) ∈ *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** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ ([[*S*]], *T*) ∈ *TRel*) ∧ *preorder Rel*
∧ *weak-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *operational-corresponding TRel* ∧ *preorder TRel* ∧ *weak-reduction-bisimulation TRel Target*
using *OC-wrt-preorder-iff-weak-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* ([[*S*]])) ∈ *Rel*)
∧ *rel-weakly-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
apply (*rule exI*) **using** *C1 C6 C7* **by** *blast*

```
  hence enc-weakly-respects-barb-set {success} ∧ 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) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TP ⟼(Calculus TWB)∗ TP′ and TP′↓<TWB>success
    hence TargetTerm TP⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
      by blast
    ultimately have TargetTerm TQ⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TQ⇓<TWB>success
        using STCalWB-reachesBarbST
      by blast
  next
    fix TP TQ TQ′
    assume (TP, TQ) ∈ TRel
    with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by simp
    moreover assume TQ ⟼(Calculus TWB)∗ TQ′ and TQ′↓<TWB>success
    hence TargetTerm TQ⇓<STCalWB SWB TWB>success
        using STCalWB-reachesBarbST
      by blast
    ultimately have TargetTerm TP⇓<STCalWB SWB TWB>success
        using C6
      by blast
    thus TP⇓<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 ∧ preorder TRel ∧ weak-reduction-bisimulation TRel Target
   ∧ enc-weakly-respects-barb-set {success} ∧ rel-weakly-respects-barb-set TRel TWB {success}
   ∧ enc-reflects-divergence ∧ 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 × ′procT) set
  shows (strongly-operational-corresponding TRel ∧ preorder TRel
          ∧ strong-reduction-bisimulation TRel Target
          ∧ enc-respects-barb-set {success} ∧ rel-respects-barb-set TRel TWB {success}
          ∧ enc-reflects-divergence ∧ rel-reflects-divergence TRel Target)
         = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
            ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
            ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⟶ (⟦S⟧, T) ∈ TRel)
            ∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
            ∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success}
            ∧ 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*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
by (*simp add*: *indRelRTPO.encR*)
from *Rel-def A8* **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 A9* **have** *B3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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 ∀ *P Q Q′*. (*P*, *Q*) ∈ *TRel* ∧ *Q* ⟼*Target Q′* ⟶ (∃ *P′*. *P* ⟼*Target P′* ∧ (*P′*, *Q′*) ∈ *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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
apply (*rule exI*) **using** *B1 B2 B3 B4 B5 B6 B7* **by** *blast*
next
assume ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*)
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*) ∧ *preorder Rel*
∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where** *C1*: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*
and *C2*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
and *C3*: ∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *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** ∃ *Rel*.(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*}
∧ (∀ *S T*. (*SourceTerm S*, *TargetTerm T*) ∈ *Rel* ⟶ (⟦*S*⟧, *T*) ∈ *TRel*) ∧ *preorder Rel*

∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
hence *strongly-operational-corresponding TRel* ∧ *preorder TRel*
∧ *strong-reduction-bisimulation TRel Target*
using *SOC-wrt-preorder-iff-strong-reduction-bisimulation*[**where** *TRel*=*TRel*]
by *simp*
moreover have ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel*)
∧ *rel-respects-barb-set Rel* (*STCalWB SWB TWB*) {*success*}
∧ *rel-reflects-divergence Rel* (*STCal Source Target*)
apply (*rule exI*) **using** *C1 C6 C7* **by** *blast*
hence *enc-respects-barb-set* {*success*} ∧ *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*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TP*↓<*TWB*>*success*
hence *TargetTerm TP*↓<*STCalWB SWB TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
ultimately have *TargetTerm TQ*↓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TQ*↓<*TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
next
fix *TP TQ*
assume (*TP*, *TQ*) ∈ *TRel*
with *C2* **have** (*TargetTerm TP*, *TargetTerm TQ*) ∈ *Rel*
by *simp*
moreover assume *TQ*↓<*TWB*>*success*
hence *TargetTerm TQ*↓<*STCalWB SWB TWB*>*success*
using *STCalWB-hasBarbST*
by *blast*
ultimately have *TargetTerm TP*↓<*STCalWB SWB TWB*>*success*
using *C6*
by *blast*
thus *TP*↓<*TWB*>*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* ∧ *preorder TRel*
∧ *strong-reduction-bisimulation TRel Target*
∧ *enc-respects-barb-set* {*success*} ∧ *rel-respects-barb-set TRel TWB* {*success*}
∧ *enc-reflects-divergence* ∧ *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 × 'procS) set
    and TRel :: ('procT × 'procT) set
  assumes eqS: equivalence SRel
      and eqT: equivalence TRel
  shows fully-abstract SRel TRel
        ∧ operational-corresponding TRel ∧ weak-reduction-bisimulation TRel Target
        ⟷ (∃ Rel.
        (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
        ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
        ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
        ∧ trans Rel ∧ 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:
   ∀ S. SourceTerm S ∼⟦·⟧T<TRel> TargetTerm (⟦S⟧) ∧ TargetTerm (⟦S⟧) ∼⟦·⟧T<TRel> SourceTerm S
    by (simp add: indRelTEQ.encR indRelTEQ.encL)
  moreover from A4 have A6: TRel = {(T1, T2). TargetTerm T1 ∼⟦·⟧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 ∼⟦·⟧T<TRel> SourceTerm S2}
  proof –
    from A6 have ∀ S1 S2. ((⟦S1⟧, ⟦S2⟧) ∈ TRel) = TargetTerm (⟦S1⟧) ∼⟦·⟧T<TRel> TargetTerm (⟦S2⟧)
      by blast
    moreover have indRelTEQ TRel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q} = indRelTEQ TRel
      by (auto simp add: indRelTEQ.encL)
    with A7 have trans (indRelTEQ TRel ∪ {(P, Q). ∃ S. ⟦S⟧ ∈T P ∧ S ∈S Q})
        unfolding trans-def
      by blast
    ultimately show SRel = {(S1, S2). SourceTerm S1 ∼⟦·⟧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 ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
        ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
        ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
        ∧ trans Rel ∧ weak-reduction-bisimulation Rel (STCal Source Target)
    by blast
next
  assume
   ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel ∧ (TargetTerm (⟦S⟧), SourceTerm S) ∈ Rel)
    ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
    ∧ trans Rel ∧ weak-reduction-bisimulation Rel (STCal Source Target)
```

from *this* **obtain** *Rel* **where**
$B1$: $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket),\ SourceTerm\ S) \in Rel$
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** $B5$: *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** $C1$: $TRel^+ = TRel$
using *trancl-id*[*of TRel*]
unfolding *equiv-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** $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
by *simp*
moreover from *B3* **have** $\forall T1\ T2.\ (T1,\ T2) \in TRel \longrightarrow (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel$
by *simp*
moreover from *B3 C1* **have** $\forall T1\ T2.\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel \longrightarrow (T1,\ T2) \in TRel^+$
by *simp*
moreover have $\forall S\ T.\ (SourceTerm\ S,\ TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket,\ T) \in TRel^*$
proof *clarify*
fix *S T*
from *B1* **have** $(TargetTerm\ (\llbracket S \rrbracket),\ SourceTerm\ S) \in Rel$
by *simp*
moreover assume $(SourceTerm\ S,\ TargetTerm\ T) \in Rel$
ultimately have $(TargetTerm\ (\llbracket S \rrbracket),\ TargetTerm\ T) \in Rel$
using *B4*
unfolding *trans-def*
by *blast*
with *B3 C2* **show** $(\llbracket S \rrbracket,\ T) \in TRel^*$
by *simp*
qed
ultimately 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-reduction-bisimulation Rel* (*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*

∧ *strong-reduction-bisimulation TRel Target* ⟷ (∃ *Rel*.
(∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} ∧ *trans Rel*
∧ *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*:
∀ *S*. *SourceTerm S* ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S*⟧) ∧ *TargetTerm* (⟦*S*⟧) ∼⟦·⟧*T*<*TRel*> *SourceTerm S*
by (*simp add*: *indRelTEQ.encR indRelTEQ.encL*)
moreover from *A4* **have** *A6*: *TRel* = {(*T1*, *T2*). *TargetTerm T1* ∼⟦·⟧*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* ∼⟦·⟧*T*<*TRel*> *SourceTerm S2*}
proof −
from *A6* **have** ∀ *S1 S2*. ((⟦*S1*⟧, ⟦*S2*⟧) ∈ *TRel*) = *TargetTerm* (⟦*S1*⟧) ∼⟦·⟧*T*<*TRel*> *TargetTerm* (⟦*S2*⟧)
by *blast*
moreover have *indRelTEQ TRel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*} = *indRelTEQ TRel*
by (*auto simp add*: *indRelTEQ.encL*)
with *A7* **have** *trans* (*indRelTEQ TRel* ∪ {(*P*, *Q*). ∃ *S*. ⟦*S*⟧ ∈*T P* ∧ *S* ∈*S Q*})
unfolding *trans-def*
by *blast*
ultimately show *SRel* = {(*S1*, *S2*). *SourceTerm S1* ∼⟦·⟧*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 ∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} ∧ *trans Rel*
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
by *blast*
next
assume
∃ *Rel*. (∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*)
∧ *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
∧ *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *Rel*} ∧ *trans Rel*
∧ *strong-reduction-bisimulation Rel* (*STCal Source Target*)
from *this* **obtain** *Rel* **where**
B1: ∀ *S*. (*SourceTerm S*, *TargetTerm* (⟦*S*⟧)) ∈ *Rel* ∧ (*TargetTerm* (⟦*S*⟧), *SourceTerm S*) ∈ *Rel*
and *B2*: *SRel* = {(*S1*, *S2*). (*SourceTerm S1*, *SourceTerm S2*) ∈ *Rel*}
and *B3*: *TRel* = {(*T1*, *T2*). (*TargetTerm T1*, *TargetTerm T2*) ∈ *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** $\forall S.\ (SourceTerm\ S,\ TargetTerm\ (\llbracket S\rrbracket)) \in Rel$
      by *simp*
    moreover from *B3* **have** $\forall T1\ T2.\ (T1,\ T2) \in TRel \longrightarrow (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel$
      by *simp*
    moreover from *B3 C1*
    have $\forall T1\ T2.\ (TargetTerm\ T1,\ TargetTerm\ T2) \in Rel \longrightarrow (T1,\ T2) \in TRel^+$
      by *simp*
    moreover have $\forall S\ T.\ (SourceTerm\ S,\ TargetTerm\ T) \in Rel \longrightarrow (\llbracket S\rrbracket,\ T) \in TRel^*$
    proof *clarify*
      fix *S T*
      from *B1* **have** $(TargetTerm\ (\llbracket S\rrbracket),\ SourceTerm\ S) \in Rel$
        by *simp*
      moreover assume $(SourceTerm\ S,\ TargetTerm\ T) \in Rel$
      ultimately have $(TargetTerm\ (\llbracket S\rrbracket),\ TargetTerm\ T) \in Rel$
          using *B4*
          unfolding *trans-def*
        by *blast*
      with *B3 C2* **show** $(\llbracket S\rrbracket,\ T) \in TRel^*$
        by *simp*
    qed
    ultimately 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*)
          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 $SP \longmapsto Source{*}\ SP'$
with *opCom* **obtain** TP' **where** $A1$: $[\![SP]\!] \longmapsto Target{*}\ TP'$ **and** $A2$: $([\![SP']\!], TP') \in TRel$
by *blast*
assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
with *bisimT A1* **obtain** TQ' **where** $A3$: $[\![SQ]\!] \longmapsto Target{*}\ TQ'$ **and** $A4$: $(TP', TQ') \in TRel$
by *blast*
from *A3 opSou* **obtain** SQ' **where** $A5$: $SQ \longmapsto Source{*}\ SQ'$ **and** $A6$: $([\![SQ']\!], TQ') \in TRel$
by *blast*
from *A2 A4 A6 symmT transT* **have** $([\![SP']\!], [\![SQ']\!]) \in TRel$
unfolding *trans-def sym-def*
by *blast*
with *fullAbs A5* **show** $\exists SQ'.\ SQ \longmapsto Source{*}\ SQ' \wedge (SP', SQ') \in SRel$
by *blast*
next
fix $SP\ SQ\ SQ'$
assume $SQ \longmapsto Source{*}\ SQ'$
with *opCom* **obtain** TQ' **where** $B1$: $[\![SQ]\!] \longmapsto Target{*}\ TQ'$ **and** $B2$: $([\![SQ']\!], TQ') \in TRel$
by *blast*
assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
with *bisimT B1* **obtain** TP' **where** $B3$: $[\![SP]\!] \longmapsto Target{*}\ TP'$ **and** $B4$: $(TP', TQ') \in TRel$
by *blast*
from *B3 opSou* **obtain** SP' **where** $B5$: $SP \longmapsto Source{*}\ SP'$ **and** $B6$: $([\![SP']\!], TP') \in TRel$
by *blast*
from *B2 B4 B6 symmT transT* **have** $([\![SP']\!], [\![SQ']\!]) \in TRel$
unfolding *trans-def sym-def*
by *blast*
with *fullAbs B5* **show** $\exists SP'.\ SP \longmapsto Source{*}\ SP' \wedge (SP', SQ') \in SRel$
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 $SP \longmapsto Source\ SP'$
with *opCom* **obtain** TP' **where** $A1$: $[\![SP]\!] \longmapsto Target\ TP'$ **and** $A2$: $([\![SP']\!], TP') \in TRel$
by *blast*
assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
with *bisimT A1* **obtain** TQ' **where** $A3$: $[\![SQ]\!] \longmapsto Target\ TQ'$ **and** $A4$: $(TP', TQ') \in TRel$
by *blast*
from *A3 opSou* **obtain** SQ' **where** $A5$: $SQ \longmapsto Source\ SQ'$ **and** $A6$: $([\![SQ']\!], TQ') \in TRel$
by *blast*
from *A2 A4 A6 symmT transT* **have** $([\![SP']\!], [\![SQ']\!]) \in TRel$
unfolding *trans-def sym-def*

by *blast*
with *fullAbs A5* **show** $\exists SQ'. SQ \longmapsto Source\ SQ' \wedge (SP', SQ') \in SRel$
by *blast*
next
fix *SP SQ SQ'*
assume $SQ \longmapsto Source\ SQ'$
with *opCom* **obtain** *TQ'* **where** *B1*: $[\![SQ]\!] \longmapsto Target\ TQ'$ **and** *B2*: $([\![SQ']\!], TQ') \in TRel$
by *blast*
assume $(SP, SQ) \in SRel$
with *fullAbs* **have** $([\![SP]\!], [\![SQ]\!]) \in TRel$
by *simp*
with *bisimT B1* **obtain** *TP'* **where** *B3*: $[\![SP]\!] \longmapsto Target\ TP'$ **and** *B4*: $(TP', TQ') \in TRel$
by *blast*
from *B3 opSou* **obtain** *SP'* **where** *B5*: $SP \longmapsto Source\ SP'$ **and** *B6*: $([\![SP']\!], TP') \in TRel$
by *blast*
from *B2 B4 B6 symmT transT* **have** $([\![SP']\!], [\![SQ']\!]) \in TRel$
unfolding *trans-def sym-def*
by *blast*
with *fullAbs B5* **show** $\exists SP'. SP \longmapsto Source\ SP' \wedge (SP', SQ') \in SRel$
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*: $\forall T. \exists S. T = [\![S]\!]$
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** $\exists S. TP = [\![S]\!]$
by *simp*
from *this* **obtain** *SP* **where** *A1*: $[\![SP]\!] = TP$
by *blast*
from *surj* **have** $\exists S. TQ = [\![S]\!]$
by *simp*
from *this* **obtain** *SQ* **where** *A2*: $[\![SQ]\!] = TQ$
by *blast*
assume $TP \longmapsto Target* \ TP'$
with *opCor A1* **obtain** *SP'* **where** *A3*: $SP \longmapsto Source*\ SP'$ **and** *A4*: $([\![SP']\!], TP') \in TRel$
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*: $SQ \longmapsto Source*\ SQ'$ **and** *A6*: $(SP', SQ') \in SRel$
by *blast*
from *opCor A2 A5* **obtain** *TQ'* **where** *A7*: $TQ \longmapsto Target*\ TQ'$ **and** *A8*: $([\![SQ']\!], TQ') \in TRel$
by *blast*
from *symmT A4* **have** $(TP', [\![SP']\!]) \in TRel$
unfolding *sym-def*
by *simp*
moreover from *fullAbs A6* **have** $([\![SP']\!], [\![SQ']\!]) \in TRel$
by *simp*
ultimately have $(TP', TQ') \in TRel$
using *transT A8*

unfolding *trans-def*
by *blast*
with *A7* **show** $\exists\, TQ'.\ TQ \longmapsto Target{*}\ TQ' \land (TP', TQ') \in TRel$
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