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.
## Contents

1 Relations 3  
1.1 Basic Conditions ........................................... 3  
1.2 Preservation, Reflection, and Respection of Predicates .......... 4  

2 Process Calculi 6  
2.1 Reduction Semantics .......................................... 6  
2.1.1 Observables or Barbs .................................... 8  

3 Simulation Relations 11  
3.1 Simulation ....................................................... 11  
3.2 Contrasimulation .............................................. 13  
3.3 Coupled Simulation .......................................... 14  
3.4 Correspondence Simulation .................................. 15  
3.5 Bisimulation .................................................. 16  
3.6 Step Closure of Relations ................................... 20  

4 Encodings 21  

5 Relation between Source and Target Terms 26  
5.1 Relations Induced by the Encoding Function .................. 26  
5.2 Relations Induced by the Encoding and a Relation on Target Terms .......... 34  
5.3 Relations Induced by the Encoding and Relations on Source Terms and Target Terms 48  

6 Success Sensitiveness and Barbs 57  

7 Divergence Reflection 60  

8 Operational Correspondence 61  
8.1 Trivial Operational Correspondence Results .................. 63  
8.2 (Strong) Operational Completeness vs (Strong) Simulation ....... 63  
8.3 Weak Operational Soundness vs Contrasimulation ............... 64  
8.4 (Strong) Operational Soundness vs (Strong) Simulation ......... 65  
8.5 Weak Operational Correspondence vs Correspondence Similarity .... 66  
8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity ...... 67  

9 Full Abstraction 69  
9.1 Trivial Full Abstraction Results ................................ 69  
9.2 Fully Abstract Encodings ..................................... 70  
9.3 Full Abstraction w.r.t. Preorders ............................. 72  
9.4 Full Abstraction w.r.t. Equivalences .......................... 74  
9.5 Full Abstraction without Relating Translations to their Source Terms .... 75  

10 Combining Criteria 77  
10.1 Divergence Reflection and Success Sensitiveness ............... 78  
10.2 Adding Operational Correspondence .......................... 78  
10.3 Full Abstraction and Operational Correspondence ............... 82
theory Relations
imports Main HOL−Library.LaTeXsugar HOL−Library.OptionalSugar
begin

1 Relations

1.1 Basic Conditions

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

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

A symmetric preorder is an equivalence.

lemma symm-preorder-is-equivalence:
fixes Rel :: (′a × ′a) set
assumes preorder Rel
and sym Rel
shows equivalence Rel
⟨proof⟩

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

definition symcl :: (′a × ′a) set ⇒ (′a × ′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 × ′a) set
and a b :: ′a
assumes elem: (a, b) ∈ Rel
shows (a, b) ∈ symcl Rel
and (b, a) ∈ symcl Rel
⟨proof⟩

The symmetric closure of a relation is symmetric.

lemma sym-symcl:
fixes Rel :: (′a × ′a) set
shows sym (symcl Rel)
⟨proof⟩

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

lemma refl-symm-closure-is-symm-refl-closure:
fixes Rel :: (′a × ′a) set
shows symcl Rel = Rel ∪ Rel−1
⟨proof⟩

The symmetric closure of a reflexive relation is reflexive.

lemma refl-symcl-of-refl-rel:
fixes Rel :: (′a × ′a) set
and A :: ′a set
assumes refl-on A Rel
shows refl-on A (symcl Rel)
⟨proof⟩

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:
The reflexive closure of a symmetric relation is symmetric.

**lemma sym-reflcl-of-symm-rel:**
- **fixes** Rel :: ('a × 'a) set
- **assumes** sym Rel
- **shows** sym (Rel^=) = symcl Rel

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

**lemma reflcl-of-refl-rel:**
- **fixes** Rel :: ('a × 'a) set
- **assumes** refl Rel
- **shows** Rel^= = Rel

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

**lemma symm-closure-of-symm-rel:**
- **fixes** Rel :: ('a × 'a) set
- **assumes** sym Rel
- **shows** symcl Rel = Rel

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

**lemma rtrancl-of-preorder:**
- **fixes** Rel :: ('a × 'a) set
- **assumes** preorder Rel
- **shows** Rel^* = Rel

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

**lemma refl-trans-closure-subset-of-refl-symm-trans-closure:**
- **fixes** Rel :: ('a × 'a) set
- **shows** Rel^* ⊆ (symcl (Rel^=))^+

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)

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 × 'a) set ⇒ ('a ⇒ bool) ⇒ bool where**
rel-preserves-pred Rel Pred ≡ ∀ a b. (a, b) ∈ Rel ∧ Pred a → Pred b
abbreviation rel-preserves-binary-pred :: ('a × 'a) set ⇒ ('a ⇒ 'b ⇒ bool) ⇒ bool where
   rel-preserves-binary-pred Rel Pred ≡ ∀ a b x. (a, b) ∈ Rel ∧ Pred a x → 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 × 'a) set ⇒ ('a ⇒ bool) ⇒ bool where
   rel-reflects-pred Rel Pred ≡ ∀ a b. (a, b) ∈ Rel ∧ Pred b → Pred a

abbreviation rel-reflects-binary-pred :: ('a × 'a) set ⇒ ('a ⇒ 'b ⇒ bool) ⇒ bool where
   rel-reflects-binary-pred Rel Pred ≡ ∀ a b x. (a, b) ∈ Rel ∧ Pred b x → Pred a x

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

abbreviation rel-respects-pred :: ('a × 'a) set ⇒ ('a ⇒ bool) ⇒ bool where
   rel-respects-pred Rel Pred ≡ rel-preserves-pred Rel Pred ∧ rel-reflects-pred Rel Pred

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

⟨ proof ⟩

lemma symm-relation-impl-preservation-equals-reflection-of-binary-predicates:
   fixes Rel :: ('a × 'a) set
   and Pred :: 'a ⇒ 'b ⇒ bool
   assumes symm: sym Rel
   shows rel-preserves-binary-pred Rel Pred = rel-reflects-binary-pred Rel Pred
   and rel-preserves-binary-pred Rel Pred = rel-reflects-binary-pred Rel Pred
   and rel-reflects-binary-pred Rel Pred = rel-respects-binary-pred Rel Pred

⟨ proof ⟩

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

lemma preservation-and-closures:
   fixes Rel :: ('a × 'a) set
   and Pred :: 'a ⇒ bool
   assumes preservation: rel-preserves-pred Rel Pred
   shows rel-preserves-pred (Rel⁺) Pred
   and rel-preserves-binary-pred (Rel⁺) Pred
   and rel-preserves-binary-pred (Rel⁺) Pred

⟨ proof ⟩

lemma preservation-of-binary-predicates-and-closures:
   fixes Rel :: ('a × 'a) set
   and Pred :: 'a ⇒ 'b ⇒ bool
   assumes preservation: rel-preserves-binary-pred Rel Pred
   shows rel-preserves-binary-pred (Rel⁺) Pred
   and rel-preserves-binary-pred (Rel⁺) Pred
   and rel-preserves-binary-pred (Rel⁺) Pred

⟨ proof ⟩

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

lemma reflection-and-closures:
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 processCalculi =
    Reductions :: 'proc ⇒ 'proc ⇒ bool

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

**Abbreviation**  
\[
\text{step :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool} \\
(\cdot \mapsto - [70, 70, 70] 80)
\]

where  
\[
P \mapsto \text{Cal} Q \equiv \text{Reductions Cal} P Q
\]

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

**Primrec**  
\[
nSteps :: 'proc ⇒ 'proc processCalculus ⇒ \text{nat} ⇒ 'proc ⇒ bool \\
(\cdot \mapsto - [70, 70, 70] 80)
\]

where  
\[
P \mapsto \text{Cal}^0 Q = (P = Q) | \\
P \mapsto \text{Cal}^{\text{Suc}} n Q = (\exists P'. P \mapsto \text{Cal}^n P' \land P' \mapsto \text{Cal} Q)
\]

**Definition**  
\[
\text{steps :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool} \\
(\cdot \mapsto - [70, 70, 70] 80)
\]

where  
\[
P \mapsto \text{Cal}^* Q \equiv \exists n. P \mapsto \text{Cal}^n Q
\]

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

**Definition**  
\[
divergent :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool \\
(\cdot \mapsto \omega [70, 70, 70] 80)
\]

where  
\[
P \mapsto (\text{Cal}^\omega) \equiv \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists P''. P' \mapsto \text{Cal} P'')
\]

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

**Lemma** steps-refl:  
\[
\text{fixes Cal :: 'proc processCalculus} \\
\text{and P :: 'proc} \\
\text{shows P \mapsto \text{Cal}^* P}
\]

(proof)

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

**Lemma** step-to-steps:  
\[
\text{fixes Cal :: 'proc processCalculus} \\
\text{and P P' :: 'proc} \\
\text{assumes step: P \mapsto \text{Cal} P'} \\
\text{shows P \mapsto \text{Cal}^* P'}
\]

(proof)

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:  
\[
\text{fixes Cal :: 'proc processCalculus} \\
\text{and n1 n2 :: nat} \\
\text{shows} \forall P Q R. P \mapsto \text{Cal}^{n1} Q \land Q \mapsto \text{Cal}^{n2} R \mapsto P \mapsto \text{Cal}^{(n1 + n2)} R
\]

(proof)

**Lemma** steps-add:  
\[
\text{fixes Cal :: 'proc processCalculus} \\
\text{and P Q R :: 'proc} \\
\text{assumes A1: P \mapsto \text{Cal}^* Q} \\
\text{and A2: Q \mapsto \text{Cal}^* R} \\
\text{shows P \mapsto \text{Cal}^* R}
\]

(proof)
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] 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] 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 (λ P 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)
  (proof)

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 (λ P 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)
  (proof)

A relation respects barbs if it preserves and reflects barbs.

abbreviation rel-respects-barb-set
abbreviation \( \text{rel-respects-barb-set} \) :: \((\text{'proc} \times \text{'proc})\text{ set} \Rightarrow (\text{'proc}, \text{'barbs}) \text{ calculusWithBarbs} \Rightarrow \text{'barbs set} \Rightarrow \text{bool}\)

where

\(\text{rel-respects-barb-set} \text{ Rel CWB Barbs} \equiv \text{rel-preserves-barb-set Rel CWB Barbs} \land \text{rel-reflects-barb-set Rel CWB Barbs}\)

abbreviation \( \text{rel-respects-barbs} \)

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

where

\(\text{rel-respects-barbs} \text{ Rel CWB} \equiv \text{rel-preserves-barbs Rel CWB} \land \text{rel-reflects-barbs Rel CWB}\)

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

fixes \(\text{Rel} :: (\text{'proc} \times \text{'proc})\text{ set}\)

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

shows \(\text{rel-respects-barbs Rel CWB} = (\forall \text{Barbs. rel-respects-barb-set Rel CWB Barbs})\)

\(\langle \text{proof} \rangle\)

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

lemma preservation-of-barbs-and-closures:

fixes \(\text{Rel} :: (\text{'proc} \times \text{'proc})\text{ set}\)

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

assumes preservation:

\(\text{rel-preserves-barbs Rel CWB}\)

shows \(\text{rel-preserves-barbs (Rel}^=\text{) CWB}\)

and \(\text{rel-preserves-barbs (Rel}^+\text{) CWB}\)

and \(\text{rel-preserves-barbs (Rel}^\ast\text{) CWB}\)

\(\langle \text{proof} \rangle\)

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

lemma reflection-of-barbs-and-closures:

fixes \(\text{Rel} :: (\text{'proc} \times \text{'proc})\text{ set}\)

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

assumes reflection:

\(\text{rel-reflects-barbs Rel CWB}\)

shows \(\text{rel-reflects-barbs (Rel}^=\text{) CWB}\)

and \(\text{rel-reflects-barbs (Rel}^+\text{) CWB}\)

and \(\text{rel-reflects-barbs (Rel}^\ast\text{) CWB}\)

\(\langle \text{proof} \rangle\)

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

lemma respection-of-barbs-and-closures:

fixes \(\text{Rel} :: (\text{'proc} \times \text{'proc})\text{ set}\)

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

assumes respection:

\(\text{rel-respects-barbs Rel CWB}\)

shows \(\text{rel-respects-barbs (Rel}^=\text{) CWB}\)

and \(\text{rel-respects-barbs (symcl Rel) CWB}\)

and \(\text{rel-respects-barbs (Rel}^+\text{) CWB}\)

and \(\text{rel-respects-barbs (symcl (Rel}^=\text{)) CWB}\)

and \(\text{rel-respects-barbs (Rel}^\ast\text{) CWB}\)

and \(\text{rel-respects-barbs ((symcl (Rel}^=\text{))}^+\text{) CWB}\)

\(\langle \text{proof} \rangle\)

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 \(\text{rel-weakly-preserves-barb-set}\)

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

where

\(\text{rel-weakly-preserves-barb-set} \text{ Rel CWB Barbs} \equiv \text{rel-preserves-binary-pred Rel (\lambda P. a. \ a \in \text{Barbs} \land P\downarrow(a < CWB)\\text{a})}\)

abbreviation \(\text{rel-weakly-preserves-barbs}\)

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

where
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

$$rel-weakly-reflects-barb-set :: (\text{'proc} \times \text{'proc}) \Rightarrow (\text{'proc}, \text{'barbs}) \Rightarrow \text{barbs set} \Rightarrow \text{bool}$$

**Where**

$$rel-weakly-reflects-barb-set \text{ Rel CWB Barbs} \equiv rel-reflects-binary-pred \text{ Rel (}\lambda P a. a \in \text{Barbs} \land P \Downarrow \text{CWB} > a\text{)}$$

**Abbreviation** rel-weakly-reflects-barbs

$$rel-weakly-reflects-barbs :: (\text{'proc} \times \text{'proc}) \Rightarrow (\text{'proc}, \text{'barbs}) \Rightarrow \text{barbs set} \Rightarrow \text{bool}$$

**Where**

$$rel-weakly-reflects-barbs \text{ Rel CWB} \equiv rel-weakly-preserves-barbs \text{ Rel CWB} \land rel-weakly-reflects-barb-set \text{ Rel CWB Barbs}$$

**Lemma** weak-reflection-of-barbs-and-set-of-barbs:

**Fixes** Rel :: (\text{'proc} \times \text{'proc}) set

**And** CWB :: ('proc, 'barbs) calculusWithBarbs

**Shows** rel-weakly-reflects-barbs Rel CWB = (\forall Barbs. rel-weakly-reflects-barb-set Rel CWB Barbs)

(\text{proof})

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

**Abbreviation** rel-weakly-respects-barb-set

$$rel-weakly-respects-barb-set :: (\text{'proc} \times \text{'proc}) \Rightarrow (\text{'proc}, \text{'barbs}) \Rightarrow \text{barbs set} \Rightarrow \text{bool}$$

**Where**

$$rel-weakly-respects-barb-set \text{ Rel CWB Barbs} \equiv rel-weakly-preserves-barb-set \text{ Rel CWB Barbs} \land rel-weakly-reflects-barb-set \text{ Rel CWB Barbs}$$

**Abbreviation** rel-weakly-respects-barbs

$$rel-weakly-respects-barbs :: (\text{'proc} \times \text{'proc}) \Rightarrow (\text{'proc}, \text{'barbs}) \Rightarrow \text{barbs set} \Rightarrow \text{bool}$$

**Where**

$$rel-weakly-respects-barbs \text{ Rel CWB} \equiv rel-weakly-preserves-barbs \text{ Rel CWB} \land rel-weakly-reflects-barbs \text{ Rel CWB}$$

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

**Fixes** Rel :: (\text{'proc} \times \text{'proc}) set

**And** CWB :: ('proc, 'barbs) calculusWithBarbs

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

(\text{proof})

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

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

**Fixes** Rel :: (\text{'proc} \times \text{'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

(\text{proof})

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

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

**Fixes** Rel :: (\text{'proc} \times \text{'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
⟨proof⟩

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 × '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⟩

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⟩
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)

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. \((P, Q) \in \text{Rel} \land P \downarrow<_{\text{CWB}} a \rightarrow Q \downarrow<_{\text{CWB}} a\)

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)

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 ≡ ∀ P Q P'. \((P, Q) \in \text{Rel} \land P \rightarrow Cal P' \rightarrow (\exists Q'. Q \rightarrow Cal Q' \land (P', Q') \in \text{Rel})\)

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

abbreviation strong-barbed-simulation
:: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
where
strong-barbed-simulation Rel CWB ≡
strong-reduction-simulation Rel (Calculus CWB) \land rel-preserves-barbs Rel CWB

A strong strong simulation is also a weak simulation.

lemma strong-impl-weak-reduction-simulation:

fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes simulation: strong-reduction-simulation Rel Cal
shows weak-reduction-simulation Rel Cal
(proof)

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

fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes simulation: strong-barbed-simulation Rel CWB
shows rel-weakly-preserves-barbs Rel CWB
(proof)
lemma strong-impl-weak-barbed-simulation:
  \begin{align*}
  \text{fixes } & \text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set} \\
  \text{and } & \text{CWB} :: (\text{'proc}, \text{'barbs}) \text{ calculusWithBarbs} \\
  \text{assumes } & \text{simulation: strong-barbed-simulation } \text{Rel} \text{ CWB} \\
  \text{shows } & \text{weak-barbed-simulation } \text{Rel} \text{ CWB} \\
  \end{align*}
\langle \text{proof} \rangle

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

lemma strong-reduction-simulation-and-closures:
  \begin{align*}
  \text{fixes } & \text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set} \\
  \text{and } & \text{Cal} :: \text{proc processCalculus} \\
  \text{assumes } & \text{simulation: strong-reduction-simulation } \text{Rel} \text{ Cal} \\
  \text{shows } & \text{strong-reduction-simulation } (\text{Rel}^-) \text{ Cal} \\
  \text{and } & \text{strong-reduction-simulation } (\text{Rel}^+) \text{ Cal} \\
  \text{and } & \text{strong-reduction-simulation } (\text{Rel}^*) \text{ Cal} \\
  \end{align*}
\langle \text{proof} \rangle

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 \text{Rel} \text{ Cal} \equiv \\
\forall P Q P'. \ (P, Q) \in \text{Rel} \land P \rightarrow\text{Cal} \rightarrow P' \rightarrow (\exists Q'. \ Q \rightarrow\text{Cal} \rightarrow Q' \land (Q', P') \in \text{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 \text{Rel} \text{ CWB} \equiv \\
weak-reduction-contrasimulation \text{Rel} (\text{Calculus CWB}) \land rel-weakly-preserves-barbs \text{Rel} \text{ CWB}

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

lemma weak-reduction-contrasimulation-and-closures:
  \begin{align*}
  \text{fixes } & \text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set} \\
  \text{and } & \text{Cal} :: \text{proc processCalculus} \\
  \text{assumes } & \text{contrasimulation: weak-reduction-contrasimulation } \text{Rel} \text{ Cal} \\
  \text{shows } & \text{weak-reduction-contrasimulation } (\text{Rel}^-) \text{ Cal} \\
  \text{and } & \text{weak-reduction-contrasimulation } (\text{Rel}^+) \text{ Cal} \\
  \text{and } & \text{weak-reduction-contrasimulation } (\text{Rel}^*) \text{ Cal} \\
  \end{align*}
\langle \text{proof} \rangle

lemma weak-barbed-contrasimulation-and-closures:
  \begin{align*}
  \text{fixes } & \text{Rel} :: (\text{'proc} \times \text{'proc}) \text{ set} \\
  \text{and } & \text{CWB} :: (\text{'proc}, \text{'barbs}) \text{ calculusWithBarbs} \\
  \text{assumes } & \text{contrasimulation: weak-barbed-contrasimulation } \text{Rel} \text{ CWB} \\
  \text{shows } & \text{weak-barbed-contrasimulation } (\text{Rel}^-) \text{ CWB} \\
  \end{align*}
and weak-barbed-contrasimulation \((\text{Rel}^+)\) \(\text{CWB}\)
and weak-barbed-contrasimulation \((\text{Rel}^*)\) \(\text{CWB}\)

\(⟨\text{proof}\rangle\)

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 \text{weak-reduction-coupled-simulation} :: (\(\text{proc} \times \text{proc}\)) set \(⇒\) \(\text{proc}\) processCalculus \(⇒\) bool
where
\begin{align*}
\text{weak-reduction-coupled-simulation} \text{Rel Cal} & ≡ \\
\forall P P', Q Q' \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \\
\rightarrow (\exists Q'. Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel}) \land (\exists Q'. Q \xrightarrow{\text{Cal}} Q' \land (Q', P') \in \text{Rel})
\end{align*}

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

abbreviation \text{weak-barbed-coupled-simulation} :: (\(\text{proc} \times \text{proc}\)) set \(⇒\) (\(\text{proc}, \text{barbs}\)) calculusWithBarbs \(⇒\) bool
where
\begin{align*}
\text{weak-barbed-coupled-simulation} \text{Rel CWB} & ≡ \\
\text{weak-reduction-coupled-simulation} \text{Rel} \text{CWB} \land \text{rel-weakly-preserves-barbs} \text{Rel CWB}
\end{align*}

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

lemma \text{weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation}:
\begin{align*}
\text{fixes} \text{Rel} :: (\text{proc} \times \text{proc})\ set \\
\text{and} \text{Cal} :: \text{\'proc} \text{processCalculus} \\
\text{shows} \text{weak-reduction-coupled-simulation} \text{Rel Cal} \\
= (\text{weak-reduction-coupled-simulation} \text{Rel Cal} \land \text{weak-reduction-contrasimulation} \text{Rel Cal})
\end{align*}
\(⟨\text{proof}\⟩\)

lemma \text{weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation}:
\begin{align*}
\text{fixes} \text{Rel} :: (\text{proc} \times \text{proc})\ set \\
\text{and} \text{CWB} :: (\text{\'proc}, \text{\'barbs}) \text{calculusWithBarbs} \\
\text{shows} \text{weak-barbed-coupled-simulation} \text{Rel CWB} \\
= (\text{weak-barbed-simulation} \text{Rel CWB} \land \text{weak-barbed-contrasimulation} \text{Rel CWB})
\end{align*}
\(⟨\text{proof}\⟩\)

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

lemma \text{weak-reduction-coupled-simulation-and-closures}:
\begin{align*}
\text{fixes} \text{Rel} :: (\text{proc} \times \text{proc})\ set \\
\text{and} \text{Cal} :: \text{\'proc} \text{processCalculus} \\
\text{assumes} \text{coupledSimulation} :: \text{weak-reduction-coupled-simulation} \text{Rel Cal} \\
\text{shows} \text{weak-reduction-coupled-simulation} \text{Rel Cal} \\
= \text{weak-reduction-coupled-simulation} \text{Rel Cal} \\
\text{and} \text{weak-reduction-coupled-simulation} \text{Rel Cal}
\end{align*}
\(⟨\text{proof}\⟩\)

lemma \text{weak-barbed-coupled-simulation-and-closures}:
\begin{align*}
\text{fixes} \text{Rel} :: (\text{proc} \times \text{proc})\ set \\
\text{and} \text{CWB} :: (\text{\'proc}, \text{\'barbs}) \text{calculusWithBarbs} \\
\text{assumes} \text{coupledSimulation} :: \text{weak-barbed-coupled-simulation} \text{Rel CWB} \\
\text{shows} \text{weak-barbed-coupled-simulation} \text{Rel CWB} \\
= \text{weak-barbed-coupled-simulation} \text{Rel CWB} \\
\text{and} \text{weak-barbed-coupled-simulation} \text{Rel CWB}
\end{align*}
\(⟨\text{proof}\⟩\)
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 R.

**abbreviation** weak-reduction-correspondence-simulation

\[
\begin{aligned}
&\text{\texttt{\langle \langle \text{proc} \times \text{proc} \rangle \set \Rightarrow \langle \text{proc processCalculus} \Rightarrow \text{bool} \rangle \text{ where} \text{ weak-reduction-correspondence-simulation } \text{Rel} \text{ Cal} \equiv} \\
&\quad (\forall P, Q) (P, Q) \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \xrightarrow{\text{Cal}} (\exists Q'). \quad Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel}) \\
&\quad \land (\forall P, Q) (P, Q) \in \text{Rel} \land Q \xrightarrow{\text{Cal}} Q' \\
&\quad \xrightarrow{\text{Cal}} (\exists P'', Q''). \quad P \xrightarrow{\text{Cal}} P'' \land Q' \xrightarrow{\text{Cal}} Q'' \land (P'', Q'') \in \text{Rel})
\end{aligned}
\]

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

**abbreviation** weak-barbed-correspondence-simulation

\[
\begin{aligned}
&\text{\texttt{\langle \langle \text{proc} \times \text{proc} \rangle \set \Rightarrow \langle \text{proc, barbs} \rangle \text{ calculusWithBarbs} \Rightarrow \text{bool} \rangle \text{ where} \text{ weak-barbed-correspondence-simulation } \text{Rel} \text{ CWB} \equiv} \\
&\quad \text{weak-reduction-correspondence-simulation} \text{Rel} \text{ (Calculus CWB)} \\
&\quad \land \text{rel-weakly-respects-barbs} \text{Rel} \text{ CWB}
\end{aligned}
\]

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 :: \((\langle \langle \text{proc} \times \text{proc} \rangle \set \Rightarrow \langle \text{proc processCalculus} \Rightarrow \langle \text{proc} \times \text{proc} \rangle \set \rangle \text{ where} \text{ cSim-cs } \text{Rel} \text{ Cal} \equiv} \\
\text{\texttt{\forall P, Q) (P, Q) \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \xrightarrow{\text{Cal}} (\exists Q'). \quad Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel})}}
\]

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

\[
\begin{aligned}
&\text{\texttt{\forall P, Q) (P, Q) \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \xrightarrow{\text{Cal}} (\exists Q'). \quad Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel})}}
\end{aligned}
\]

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

\[
\begin{aligned}
&\text{\texttt{\forall P, Q) (P, Q) \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \xrightarrow{\text{Cal}} (\exists Q'). \quad Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel})}}
\end{aligned}
\]

**lemma** reduction-correspondence-simulation-condition-trans:

\[
\begin{aligned}
&\text{\texttt{\forall P, Q) (P, Q) \in \text{Rel} \land P \xrightarrow{\text{Cal}} P' \xrightarrow{\text{Cal}} (\exists Q'). \quad Q \xrightarrow{\text{Cal}} Q' \land (P', Q') \in \text{Rel})}}
\end{aligned}
\]
The reflexive and/or transitive closure of a weak correspondence simulation is a weak correspondence simulation.

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

```plaintext
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**

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

```plaintext
fixes Rel :: ('proc × '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**

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

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

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

**abbreviation** weak-barbed-bisimulation

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

A symmetric weak simulation is a weak bisimulation.

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

```plaintext
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes sym Rel
and weak-reduction-simulation Rel Cal
shows weak-reduction-bisimulation Rel Cal
```

**proof**

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

```plaintext
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes sym Rel
and weak-barbed-simulation Rel Cal
shows weak-barbed-bisimulation Rel Cal
```

**proof**

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 × '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⟩

lemma weak-reduction-bisimulations-impl-inverse-is-simulation:
  fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
  assumes bisim: weak-reduction-bisimulation Rel Cal
  shows weak-reduction-simulation (Rel\(^{-1}\)) Cal
⟨proof⟩

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
⟨proof⟩

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⟩

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⟩

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
⟨proof⟩

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⟩

lemma weak-barbed-bisimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes bisim: weak-barbed-bisimulation Rel CWB
shows weak-barbed-bisimulation (Rel⁻¹) CWB
and weak-barbed-bisimulation (symcl Rel) CWB
and weak-barbed-bisimulation (Rel⁺) CWB
and weak-barbed-bisimulation (symcl (Rel⁻¹)) CWB
and weak-barbed-bisimulation (Rel∗) CWB
and weak-barbed-bisimulation ((symcl (Rel⁻¹))⁺) CWB
⟨proof⟩

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 × ′proc) set ⇒ ′proc processCalculus ⇒ bool
where
strong-reduction-bisimulation Rel Cal ≡
∀ P Q P', (P, Q) ∈ Rel ∧ P "→" Cal P' → (∃ Q'. Q "→" Cal Q' ∧ (P', Q') ∈ Rel)
∧ (∀ P Q Q'. (P, Q) ∈ Rel ∧ Q "→" Cal Q' → (∃ P'. P "→" Cal P' ∧ (P', Q') ∈ Rel))

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

abbreviation strong-barbed-bisimulation
:: (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ bool
where
strong-barbed-bisimulation Rel CWB ≡
strong-reduction-bisimulation Rel (Calculus CWB) ∧ rel-respects-barbs Rel CWB

A symmetric strong simulation is a strong bisimulation.

lemma symm-strong-reduction-simulation-is-bisimulation:
fixes Rel :: (′proc × ′proc) set
and Cal :: ′proc processCalculus
assumes sym Rel
and strong-reduction-simulation Rel Cal
shows strong-reduction-bisimulation Rel Cal
⟨proof⟩

lemma symm-strong-barbed-simulation-is-bisimulation:
fixes Rel :: (′proc × ′proc) set
and CWB :: (′proc, ′barbs) calculusWithBarbs
assumes sym Rel
and strong-barbed-simulation Rel CWB
shows strong-barbed-bisimulation Rel CWB
⟨proof⟩

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 × ′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⟩

lemma strong-reduction-bisimulations-impl-inverse-is-simulation:
fixes Rel :: (′proc × ′proc) set
and Cal :: ′proc processCalculus
assumes bisim: strong-reduction-bisimulation Rel Cal
shows strong-reduction-simulation (Rel⁻¹) Cal
⟨proof⟩
lemma strong-reduction-simulations-iff-bisimulation:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
shows \(\text{strong-reduction-simulation } Rel \ Cal \land \text{strong-reduction-simulation } (Rel^{-1}) \ Cal\) = \(\text{strong-reduction-bisimulation } Rel \ Cal\)
⟨proof⟩

lemma strong-barbed-simulations-iff-bisimulation:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
shows \(\text{strong-barbed-simulation } Rel \ CWB \land \text{strong-barbed-simulation } (Rel^{-1}) \ CWB\) = \(\text{strong-barbed-bisimulation } Rel \ CWB\)
⟨proof⟩

A strong bisimulation is a weak bisimulation.

lemma strong-impl-weak-reduction-bisimulation:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes bisim: \(\text{strong-reduction-bisimulation } Rel \ Cal\)
shows \(\text{weak-reduction-bisimulation } Rel \ Cal\)
⟨proof⟩

lemma strong-barbed-bisimulation-impl-weak-respection-of-barbs:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes bisim: \(\text{strong-barbed-bisimulation } Rel \ CWB\)
shows \(\text{rel-weakly-respects-barbs } Rel \ CWB\)
⟨proof⟩

lemma strong-impl-weak-barbed-bisimulation:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes bisim: \(\text{strong-barbed-bisimulation } Rel \ CWB\)
shows \(\text{weak-barbed-bisimulation } Rel \ CWB\)
⟨proof⟩

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: \(\text{strong-reduction-bisimulation } Rel \ Cal\)
shows \(\text{strong-reduction-bisimulation } (Rel^=) \ Cal\)
and \(\text{strong-reduction-bisimulation } (symcl Rel) \ Cal\)
and \(\text{strong-reduction-bisimulation } (Rel^+) \ Cal\)
and \(\text{strong-reduction-bisimulation } ((symcl (Rel^=))^+) \ Cal\)
⟨proof⟩

lemma strong-barbed-bisimulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes bisim: \(\text{strong-barbed-bisimulation } Rel \ CWB\)
shows \(\text{strong-barbed-bisimulation } (Rel^=) \ CWB\)
and \(\text{strong-barbed-bisimulation } (symcl Rel) \ CWB\)
and \(\text{strong-barbed-bisimulation } (Rel^+) \ CWB\)
and \(\text{strong-barbed-bisimulation } ((symcl (Rel^=))^+) \ CWB\)
⟨proof⟩
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.

\textbf{inductive-set} \texttt{stepsClosure} :: ('a × 'a) set ⇒ 'a processCalculus ⇒ ('a × 'a) set

\texttt{for} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{where}
\texttt{rel:} \ (P, Q) ∈ Rel ⇒ (P, Q) ∈ stepsClosure Rel Cal
\texttt{steps:} \ P ←→ Cal s P' ⇒ (P', P) ∈ stepsClosure Rel Cal
\texttt{trans:} \ [(P, Q) ∈ stepsClosure Rel Cal; (Q, R) ∈ stepsClosure Rel Cal] ⇒ (P, R) ∈ stepsClosure Rel Cal

\textbf{abbreviation} \texttt{stepsClosureInfix} ::
\texttt{′a ⇒ (′a × ′a) set ⇒ ′a processCalculus ⇒ ′a ⇒ bool (- ℝ→<,−> - [75, 75, 75, 75] 80)}
\texttt{where}
\texttt{P RPM⇒<Rel,Cal> Q} ⇒ (P, Q) ∈ stepsClosure Rel Cal

Applying the steps closure twice does not change the relation.

\textbf{lemma} \texttt{steps-closure-of-steps-closure}:
\texttt{fixes} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{shows} \texttt{stepsClosure (stepsClosure Rel Cal) Cal = stepsClosure Rel Cal}
\texttt{⟨proof⟩}

The steps closure is a preorder.

\textbf{lemma} \texttt{stepsClosure-refl}:
\texttt{fixes} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{shows} \texttt{refl (stepsClosure Rel Cal)}
\texttt{⟨proof⟩}

\textbf{lemma} \texttt{refl-trans-closure-of-rel-impl-steps-closure}:
\texttt{fixes} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{and} \ P Q :: 'a
\texttt{assumes} \texttt{(P, Q) ∈ Rel*}
\texttt{shows} \texttt{P RPM⇒<Rel,Cal> Q}
\texttt{⟨proof⟩}

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

\textbf{lemma} \texttt{steps-closure-is-weak-reduction-simulation}:
\texttt{fixes} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{shows} \texttt{weak-reduction-simulation (stepsClosure Rel Cal) Cal}
\texttt{⟨proof⟩}

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

\textbf{lemma} \texttt{inverse-contrasimulation-impl-reverse-pair-in-steps-closure}:
\texttt{fixes} \ Rel :: ('a × 'a) set
\texttt{and} \ Cal :: 'a processCalculus
\texttt{and} \ P Q :: 'a
\texttt{assumes} \texttt{con: weak-reduction-contrasimulation (Rel⁻¹) Cal}
\texttt{and} \texttt{pair:} \texttt{(P, Q) ∈ Rel}
\texttt{shows} \texttt{Q RPM⇒<Rel,Cal> P}
\texttt{⟨proof⟩}

\textbf{lemma} \texttt{simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation}:
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 × '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 
(proof)

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

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

fixes Rel :: ('a × 'a) set 
and Cal :: 'a processCalculus 
and P Q :: 'a
assumes A1: P R↦→<Rel,Cal> Q 
and A2: weak-reduction-simulation Rel Cal 
shows ∃ Q'. Q ↦→Cal* Q' ∧ (P, Q') ∈ Rel* 
(proof)

**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⁻¹,Cal> Q 
and A2: weak-reduction-contrasimulation (Rel⁻¹) Cal 
shows ∃ Q'. Q ↦→Cal* Q' ∧ (Q', P) ∈ Rel* 
(proof)

**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 ∃ Q'. Q ↦→Cal* Q' ∧ (P, Q') ∈ Rel* 
(proof)

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 ⇒ 'procT processCalculus 
  ⇒ (('procS, 'procT) Proc) processCalculus 
  where 
  STCal Source Target ⊑ 
  (Reductions = λP. (∃SP SP. P = SourceTerm SP ∧ P’ = SourceTerm SP’ ∧ Reduncions Source SP SP) ∨ 
  (∃TP TP. P = TargetTerm TP ∧ P’ = TargetTerm TP’ ∧ Reduncions Target TP TP))

definition STCalWB :: ('procS, 'barbs) calculusWithBarbs ⇒ ('procT, 'barbs) calculusWithBarbs 
  ⇒ (('procS, 'procT) Proc, 'barbs) calculusWithBarbs 
  where 
  STCalWB Source Target ⊑ 
  (Calculus = STCal (calculusWithBarbs.Calculus Source) (calculusWithBarbs.Calculus Target), 
  HasBarb = λP a. (∃SP. P = SourceTerm SP ∧ (calculusWithBarbs.HasBarb Source) SP a) ∨ 
  (∃TP. P = TargetTerm TP ∧ (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 ⇒ 'procT 
begin

abbreviation enc :: 'procS ⇒ 'procT ([1] [65] 70) where 
  [S] ⊑ Enc S 

abbreviation isSource :: ('procS, 'procT) Proc ⇒ bool (- ∈ ProcS [70] 80) where 
  P ∈ ProcS ⊑ (∃S. P = SourceTerm S)

abbreviation isTarget :: ('procS, 'procT) Proc ⇒ bool (- ∈ ProcT [70] 80) where 
  P ∈ ProcT ⊑ (∃T. P = TargetTerm T)

abbreviation getSource :: 'procS ⇒ ('procS, 'procT) Proc ⇒ bool (- ∈S - [70, 70] 80) 
  where 
  S ∈S P ⊑ (P = SourceTerm S)

abbreviation getTarget :: 'procT ⇒ ('procS, 'procT) Proc ⇒ bool (- ∈T - [70, 70] 80) 
  where 
  T ∈T P ⊑ (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 ⇒ ('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 stepST-STCal-step: 
  fixes P P’ :: ('procS, 'procT) Proc 
  shows P →(STCal Source Target) P’ = P →ST P’ 
  (proof)
lemma $ST_{\text{Step-step}}$:
fixes $S$ :: 'proc$S$
and $T$ :: 'proc$T$
and $P'$ :: ('proc$S$, 'proc$T$) Proc
shows $\text{SourceTerm } S \mapsto ST P' = (\exists S' \cdot S' \in S P' \land S \mapsto \text{Source } S')$
and $\text{TargetTerm } T \mapsto ST P' = (\exists T' \cdot T' \in T P' \land T \mapsto \text{Target } T')$
(proof)

lemma $ST_{\text{Cal-step}}$:
fixes $S$ :: 'proc$S$
and $T$ :: 'proc$T$
and $P'$ :: ('proc$S$, 'proc$T$) Proc
shows $\text{SourceTerm } S \mapsto (ST_{\text{Cal Source Target}}) P' = (\exists S' \cdot S' \in S P' \land S \mapsto \text{Source } S')$
and $\text{TargetTerm } T \mapsto (ST_{\text{Cal Source Target}}) P' = (\exists T' \cdot T' \in T P' \land T \mapsto \text{Target } T')$
(proof)

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 $\text{stepsST}$
:: ('proc$S$, 'proc$T$) Proc \Rightarrow ('proc$S$, 'proc$T$) Proc \Rightarrow bool [\mapsto ST* \in [70, 70] 80]
where
$P \mapsto ST* P' \equiv$
$(\exists S S'. S \in S P \land S' \in S P' \land S \mapsto \text{Source } S') \lor (\exists T T'. T \in T P \land T' \in T P' \land T \mapsto \text{Target* } T')$

lemma $ST_{\text{Steps-steps}}$:
fixes $S$ :: 'proc$S$
and $T$ :: 'proc$T$
and $P'$ :: ('proc$S$, 'proc$T$) Proc
shows $\text{SourceTerm } S \mapsto ST* P' = (\exists S' \cdot S' \in S P' \land S \mapsto \text{Source* } S')$
and $\text{TargetTerm } T \mapsto ST* P' = (\exists T' \cdot T' \in T P' \land T \mapsto \text{Target* } T')$
(proof)

lemma $ST_{\text{Cal-steps}}$:
fixes $S$ :: 'proc$S$
and $T$ :: 'proc$T$
and $P'$ :: ('proc$S$, 'proc$T$) Proc
shows $\text{SourceTerm } S \mapsto (ST_{\text{Cal Source Target}})* P' = (\exists S' \cdot S' \in S P' \land S \mapsto \text{Source* } S')$
and $\text{TargetTerm } T \mapsto (ST_{\text{Cal Source Target}})* P' = (\exists T' \cdot T' \in T P' \land T \mapsto \text{Target* } T')$
(proof)

lemma $\text{stepsST-}ST_{\text{Cal-steps}}$:
fixes $P$ :: ('proc$S$, 'proc$T$) Proc
shows $P \mapsto (ST_{\text{Cal Source Target}})* P' = P \mapsto ST* P'$
(proof)

lemma $\text{stepsST-refl}$:
fixes $P$ :: ('proc$S$, 'proc$T$) Proc
shows $P \mapsto ST* P$
(proof)

lemma $\text{stepsST-add}$:
fixes $P$ :: ('proc$S$, 'proc$T$) Proc
assumes $A1$: $P \mapsto ST* Q$
and $A2$: $Q \mapsto ST* R$
shows $P \mapsto ST* R$
(proof)

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

abbreviation $\text{divergentST}$
\[
\begin{align*}
\text{proc} S & \quad \text{proc} T \\
\text{Proc} & \Rightarrow \text{bool} \\
\text{ST} & \omega [70, 80]
\end{align*}
\]

\[
P \rightarrow \text{ST} \omega \equiv (\exists S. S \in S \land S \rightarrow (\text{Source})\omega) \lor (\exists T. T \in T \land T \rightarrow (\text{Target})\omega)
\]

**Lemma** \(\text{STCal-divergent}:

- **fixes** \(S :: \text{proc} S \)
- **and** \(T :: \text{proc} T \)
- **shows** \(\text{SourceTerm} S \rightarrow (\text{STCal Source Target})\omega = S \rightarrow (\text{Source})\omega\)
- **and** \(\text{TargetTerm} T \rightarrow (\text{STCal Source Target})\omega = T \rightarrow (\text{Target})\omega\)

**Proof**

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 \(P(\text{enc} S)\) for all source terms \(S\).

**Abbreviation** \(\text{enc-preserves-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-preserves-pred} \equiv \forall S. \text{Pred} \rightarrow (\text{TargetTerm} ([S]))
\]

**Abbreviation** \(\text{enc-preserves-binary-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-preserves-binary-pred} \equiv \forall S x. \text{Pred} \rightarrow (\text{TargetTerm} ([S]) x)
\]

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

**Abbreviation** \(\text{enc-reflects-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-reflects-pred} \equiv \forall \text{TargetTerm} ([S]) \rightarrow (\text{SourceTerm} S)
\]

**Abbreviation** \(\text{enc-reflects-binary-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-reflects-binary-pred} \equiv \forall \text{TargetTerm} ([S]) x \rightarrow (\text{SourceTerm} S) x
\]

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

**Abbreviation** \(\text{enc-respects-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-respects-pred} \equiv \text{enc-preserves-pred} \land \text{enc-reflects-pred}
\]

**Abbreviation** \(\text{enc-respects-binary-pred} :: (((\text{proc} S, \text{proc} T) \text{Proc} \Rightarrow 'b \Rightarrow \text{bool}) \Rightarrow \text{bool})\) where

\[
\text{enc-respects-binary-pred} \equiv \text{enc-preserves-binary-pred} \land \text{enc-reflects-binary-pred}
\]

**End**

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

**Locale** \(\text{encoding-wrt-barbs} = \)

**fixes** \(\text{SWB} :: (\text{proc} S, \text{barbs}) \text{calculusWithBarbs} \)

**and** \(\text{TWB} :: (\text{proc} T, \text{barbs}) \text{calculusWithBarbs} \)

**Assumes** \(\text{calS} : \text{calculusWithBarbs}, \text{calculus SWB} = \text{Source} \)

**and** \(\text{calT} : \text{calculusWithBarbs}, \text{calculus TWB} = \text{Target} \)

**Begin**
lemma \texttt{STCalWB-STCal}:
\texttt{shows} Calculus (\texttt{STCalWB SWB TWB}) = \texttt{STCal Source Target}

\texttt{proof}

We say a term \( P \) of \texttt{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 \texttt{hasBarbST}:
\[ (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \Rightarrow 'barbs \Rightarrow \texttt{bool} \ (\exists S. S \in \texttt{P} \land S \downarrow <\texttt{SWB}>a) \lor (\exists T. T \in \texttt{P} \land T \downarrow <\texttt{TWB}>a) \]

lemma \texttt{STCalWB-hasBarbST}:
\texttt{fixes} \( P :: (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \\texttt{and} \ a :: 'barbs \texttt{shows} P \downarrow <\texttt{STCalWB SWB TWB}>a = P \downarrow .a \texttt{proof} \)

lemma \texttt{preservation-of-barbs-in-barbed-encoding}:
\texttt{fixes} \( \texttt{Rel} :: ((\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \times (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc}) \texttt{set} \\texttt{and} \ P Q :: (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \\texttt{and} \ a :: 'barbs \texttt{assumes} preservation: \texttt{rel-preserves-barbs} \texttt{Rel} (\texttt{STCalWB SWB TWB}) \\texttt{and} \ rel: (P, Q) \in \texttt{Rel} \\texttt{and} \ barb: P \downarrow .a \texttt{shows} Q \downarrow .a \texttt{proof} \)

lemma \texttt{reflection-of-barbs-in-barbed-encoding}:
\texttt{fixes} \( \texttt{Rel} :: ((\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \times (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc}) \texttt{set} \\texttt{and} \ P Q :: (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \\texttt{and} \ a :: 'barbs \texttt{assumes} reflection: \texttt{rel-reflects-barbs} \texttt{Rel} (\texttt{STCalWB SWB TWB}) \\texttt{and} \ rel: (P, Q) \in \texttt{Rel} \\texttt{and} \ barb: Q \downarrow .a \texttt{shows} P \downarrow .a \texttt{proof} \)

lemma \texttt{respection-of-barbs-in-barbed-encoding}:
\texttt{fixes} \( \texttt{Rel} :: ((\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \times (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc}) \texttt{set} \\texttt{and} \ P Q :: (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \\texttt{and} \ a :: 'barbs \texttt{assumes} respection: \texttt{rel-respects-barbs} \texttt{Rel} (\texttt{STCalWB SWB TWB}) \\texttt{and} \ rel: (P, Q) \in \texttt{Rel} \texttt{shows} P \downarrow .a = Q \downarrow .a \texttt{proof} \)

A term \( P \) of \texttt{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 \texttt{reachesBarbST}:
\[ (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \Rightarrow 'barbs \Rightarrow \texttt{bool} \ (\exists S. S \in \texttt{P} \land S \downarrow <\texttt{SWB}>a) \lor (\exists T. T \in \texttt{P} \land T \downarrow <\texttt{TWB}>a) \]

lemma \texttt{STCalWB-reachesBarbST}:
\texttt{fixes} \( P :: (\langle \texttt{procS}, \texttt{procT} \rangle \texttt{Proc} \\texttt{and} \ a :: 'barbs \texttt{shows} P \downarrow <\texttt{STCalWB SWB TWB}>a = P \downarrow .a \texttt{proof} \)

25
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)

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)

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)

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
cencR: (SourceTerm S, TargetTerm ([S])) ∈ indRelR

abbreviation (in encoding) indRelRinfix ::
  ('procS, 'procT) Proc ⇒ ('procS, 'procT) Proc ⇒ bool (- R[.]R - [75, 75, 80])
  where
  P R[.]R Q ≜ (P, Q) ∈ indRelR

inductive-set (in encoding) indRelRPO :: ((('procS, 'procT) Proc × ('procS, 'procT) Proc)) set
  where
cencR: (SourceTerm S, TargetTerm ([S])) ∈ indRelRPO |
ssource: (SourceTerm S, SourceTerm S) ∈ indRelRPO |
target: $(\text{TargetTerm} \ T, \ \text{TargetTerm} \ T) \in \text{indRelRPO} |$
trans: $[(P, Q) \in \text{indRelRPO}; (Q, R) \in \text{indRelRPO}] \implies (P, R) \in \text{indRelRPO}$

**abbreviation (in encoding)** $\text{indRelRPOinfix} ::$

`('procS, 'procT) Proc ⇒ ('procS, 'procT) Proc ⇒ bool (- ≤ [\mathit{R} - \lfloor 75, 75 \rfloor 80)`

where $P ≤ [\mathit{R} Q \equiv (P, Q) \in \text{indRelRPO}$

**lemma (in encoding)** $\text{indRelRPO-refl:}$

shows refl $\text{indRelRPO}$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{indRelRPO-is-preorder:}$

shows preorder $\text{indRelRPO}$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{refl-trans-closure-of-indRelR:}$

shows $\text{indRelRPO} = \text{indRelR}^*$

(\text{\textit{proof}})

The relation $\text{indRelR}$ 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 $\text{indRelR}$.

**lemma (in encoding)** $\text{indRelR-impl-exists-source-target-relation:}$

\text{\textit{fixes}} $\text{PredA} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \text{ set ⇒ bool}$
and $\text{PredB} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \text{ ⇒ bool}$
shows $\text{PredA indRelR} \implies \exists \text{ Rel. } (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel} \land \text{PredA Rel}$
and $(\forall (P, Q) \in \text{indRelR}. \ \text{PredB} (P, Q)$
$\implies \exists \text{ Rel. } (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\forall (P, Q) \in \text{Rel.} \ \text{PredB} (P, Q))$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{source-target-relation-impl-indRelR:}$

\text{\textit{fixes}} $\text{Rel} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \text{ set}$
and $\text{Pred} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \text{ ⇒ bool}$
assumes $\text{encRRel} : \forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel}$
and $\text{condRel} : \forall (P, Q) \in \text{Rel.} \ \text{Pred} (P, Q)$
shows $\forall (P, Q) \in \text{indRelR}. \ \text{Pred} (P, Q)$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{indRelR-iff-exists-source-target-relation:}$

\text{\textit{fixes}} $\text{Pred} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \text{ ⇒ bool}$
shows $(\forall (P, Q) \in \text{indRelR} \land \text{Pred} (P, Q))$
$\equiv (\exists \text{ Rel. } (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\forall (P, Q) \in \text{Rel} \ \text{Pred} (P, Q)))$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{indRelR-modulo-pred-impl-indRelRPO-modulo-pred:}$

\text{\textit{fixes}} $\text{Pred} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \Rightarrow bool$
assumes $\text{reflCond: } \forall P. \ \text{Pred} (P, P)$
and $\text{transCond: } \forall P \ Q \ R. \ \text{Pred} (P, Q) \land \text{Pred} (Q, R) \rightarrow \text{Pred} (P, R)$
shows $(\forall (P, Q) \in \text{indRelR} \land \text{Pred} (P, Q)) = (\forall (P, Q) \in \text{indRelRPO} \land \text{Pred} (P, Q))$

(\text{\textit{proof}})

**lemma (in encoding)** $\text{indRelRPO-iff-exists-source-target-relation:}$

\text{\textit{fixes}} $\text{Pred} :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) \Rightarrow bool$
shows $(\forall (P, Q) \in \text{indRelRPO} \land \text{Pred} (P, Q)) = (\exists \text{ Rel. } (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel}$
$\land (\forall (P, Q) \in \text{Rel} \ \text{Pred} (P, Q)) \land \text{preorder Rel})$

(\text{\textit{proof}})

An encoding preserves, reflects, or respects a predicate iff $\text{indRelR}$ 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]))
sshows ∀ (P, Q) ∈ indRelR. Pred (P, Q)
⟨proof⟩

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)
sshows ∀ S. Pred (SourceTerm S, TargetTerm ([S]))
⟨proof⟩

lemma (in encoding) enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred:
fixes Pred :: (’procS, ’procT) Proc × (’procS, ’procT) Proc ⇒ bool
shows ∀ S. Pred (SourceTerm S, TargetTerm ([S])) a = (∀ (P, Q) ∈ indRelR. ∀ a. Pred (P, Q) a)
⟨proof⟩

lemma (in encoding) enc-preserves-pred-iff-indRelR-preserves-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-preserves-pred Pred = rel-preserves-pred indRelR Pred
⟨proof⟩

lemma (in encoding) enc-preserves-binary-pred-iff-indRelR-preserves-binary-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-preserves-binary-pred Pred = rel-preserves-binary-pred indRelR Pred
⟨proof⟩

lemma (in encoding) enc-preserves-pred-iff-indRelRPO-preserves-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-preserves-pred Pred = rel-preserves-pred indRelRPO Pred
⟨proof⟩

lemma (in encoding) enc-reflects-pred-iff-indRelR-reflects-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-reflects-pred Pred = rel-reflects-pred indRelR Pred
⟨proof⟩

lemma (in encoding) enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-reflects-binary-pred Pred = rel-reflects-binary-pred indRelR Pred
⟨proof⟩

lemma (in encoding) enc-respects-pred-iff-indRelR-respects-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-respects-pred Pred = rel-respects-pred indRelR Pred
⟨proof⟩

lemma (in encoding) enc-respects-binary-pred-iff-indRelR-respects-binary-pred:
fixes Pred :: (’procS, ’procT) Proc ⇒ bool
shows enc-respects-binary-pred Pred = rel-respects-binary-pred indRelR Pred
⟨proof⟩
To analyse the reflection of source term behaviours we use relations that contain the pairs \((\text{enc } S, S)\).

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) \(\text{enc-respects-pred-iff-indRelRPO-respects-pred}:\)

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\text{enc-respects-pred }\text{Pred} = \text{ rel-respects-pred indRelRPO }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-respects-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-preserves-binary-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-preserves-binary-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-reflects-binary-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-reflects-binary-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-respects-pred Rel }\text{Pred}\)

(proof)

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

**fixes** \(\text{Pred} ::= (\text{proc } S, \text{proc } T) \text{ Proc } \Rightarrow \text{ bool}\)

**shows** \(\forall S. \text{Pred} (\text{SourceTerm } S, \text{TargetTerm } ([S])) \Rightarrow \exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \land \text{rel-respects-binary-pred Rel }\text{Pred}\)

(proof)

To analyse the reflection of source term behaviours we use relations that contain the pairs \((\text{enc } S, S)\) for all source terms \(S\).
inductive-set (in encoding) indRelL
  :: (("procS", "procT) Proc × ("procS", "procT) Proc) set
  where
ever: (TargetTerm ([S]), SourceTerm S) ∈ indRelL

abbreviation (in encoding) indRelLinfix ::
  ("procS", "procT) Proc ⇒ ("procS", "procT) Proc ⇒ bool (- R[.]L - [75, 75] 80)
  where
  P R[.]L Q ≡ (P, Q) ∈ indRelL

inductive-set (in encoding) indRelLPO
  :: (("procS", "procT) Proc × ("procS", "procT) Proc) set
  where
ever: (TargetTerm ([S]), SourceTerm S) ∈ indRelLPO |
  source: (SourceTerm S, SourceTerm S) ∈ indRelLPO |
  target: (TargetTerm T, TargetTerm T) ∈ indRelLPO |
  trans: [(P, Q) ∈ indRelLPO; (Q, R) ∈ indRelLPO] ⇒ (P, R) ∈ indRelLPO

abbreviation (in encoding) indRelLPOinfix ::
  ("procS", "procT) Proc ⇒ ("procS", "procT) Proc ⇒ bool (- ≲[.]L - [75, 75] 80)
  where
  P ≲[.]L Q ≡ (P, Q) ∈ indRelLPO

lemma (in encoding) indRelLPO-refl:
  shows refl indRelLPO
  ⟨proof⟩

lemma (in encoding) indRelLPO-is-preorder:
  shows preorder indRelLPO
  ⟨proof⟩

lemma (in encoding) refl-trans-closure-of-indRelL:
  shows indRelLPO = indRelL*
  ⟨proof⟩

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 ⇒ bool
  shows rel-preserves-pred indRelR Pred = rel-reflects-pred indRelL Pred
  ⟨proof⟩

lemma (in encoding) indRelR-preserves-binary-pred-iff-indRelL-reflects-binary-pred:
  fixes Pred :: ("procS", "procT) Proc ⇒ "b ⇒ bool
  shows rel-preserves-binary-pred indRelR Pred = rel-reflects-binary-pred indRelL Pred
  ⟨proof⟩

lemma (in encoding) indRelR-reflects-pred-iff-indRelL-preserves-pred:
  fixes Pred :: ("procS", "procT) Proc ⇒ bool
  shows rel-reflects-pred indRelR Pred = rel-preserves-pred indRelL Pred
  ⟨proof⟩

lemma (in encoding) indRelR-reflects-binary-pred-iff-indRelL-preserves-binary-pred:
  fixes Pred :: ("procS", "procT) Proc ⇒ "b ⇒ bool
  shows rel-reflects-binary-pred indRelR Pred = rel-preserves-binary-pred indRelL Pred
  ⟨proof⟩

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
  ⟨proof⟩

lemma (in encoding) refl-trans-closure-of-indRelL:
  shows indRelLPO = indRelL*
  ⟨proof⟩
shows \( \text{rel-respects-pred indRelR \ Pred} = \text{rel-respects-pred indRelL \ Pred} \)

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-respects-binary-pred-iff-indRelL-respects-binary-pred:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ rel-respects-binary-pred indRelR \ Pred} = \text{rel-respects-binary-pred indRelL \ Pred} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-preservation-iff-indRelL-cond-reflection:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\})) \in \text{Rel}) \land \text{rel-preserves-binary-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-reflects-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-binary-preservation-iff-indRelL-cond-binary-reflection:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\})) \in \text{Rel}) \land \text{rel-preserves-binary-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-reflects-binary-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-reflection-iff-indRelL-cond-preservation:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\)) \in \text{Rel}) \land \text{rel-reflects-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-preserves-binary-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-binary-reflection-iff-indRelL-cond-binary-preservation:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\))) \in \text{Rel}) \land \text{rel-reflects-binary-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-preserves-binary-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-respection-iff-indRelL-cond-respection:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\))) \in \text{Rel}) \land \text{rel-respects-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-respects-binary-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

**lemma (in encoding) indRelR-cond-binary-respection-iff-indRelL-cond-binary-respection:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool} \]

\[ \text{shows \ (3 \ Rel. \ (\forall \ S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S]\))) \in \text{Rel}) \land \text{rel-respects-binary-pred Rel \ Pred})} \]

\[ = (3 \ Rel. \ (\forall \ S. \ (\text{TargetTerm} \ ([S]\)), \ \text{SourceTerm} \ S) \in \text{Rel}) \land \text{rel-respects-binary-pred Rel \ Pred})} \]

\langle \text{proof} \rangle

An encoding preserves, reflects, or respects a predicate iff \( \text{indRelL-reflects} \) reflects, preserves, or respects this predicate.

**lemma (in encoding) enc-preserves-pred-iff-indRelL-reflects-pred:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow \text{bool} \]

\[ \text{shows \ enc-preserves-pred \ Pred} = \text{rel-reflects-pred \ indRelL \ Pred} \]

\langle \text{proof} \rangle

**lemma (in encoding) enc-reflects-pred-iff-indRelL-preserves-pred:**

\[ \text{fixes \ Pred} :: \ ('\text{procS}', '\text{procT}') \text{ Proc} \Rightarrow \text{bool} \]

\[ \text{shows \ enc-reflects-pred \ Pred} = \text{rel-preserves-pred \ indRelL \ Pred} \]

\langle \text{proof} \rangle

**lemma (in encoding) enc-respects-pred-iff-indRelL-respects-pred:**
An encoding preserves, reflects, or respects a predicate if 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-preserves-pred:**

**Fixes** $Pred :: (\text{'}procS, \text{'}procT) \text{ Proc } \Rightarrow \text{ bool}$

**Shows** $\text{enc-preserves-pred } Pred = \text{ rel-preserves-pred indRelL } Pred$

\[ \langle \text{proof} \rangle \]

To analyse the respect of source term behaviours we use relations that contain both kind of pairs: $(S, \text{enc } S)$ as well as $(\text{enc } S, S)$ for all source terms $S$.

**Inductive-set (in encoding) indRel**

\[ \text{indRel : } (((\text{'}procS, \text{'}procT) \text{ Proc}) \times ((\text{'}procS, \text{'}procT) \text{ Proc})) \text{ set} \]

**Where**

\begin{align*}
\text{encR} & : (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRel} | \\
\text{encL} & : (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{indRel}
\end{align*}

**Abbreviation (in encoding) indRelInfix**

\[ (\text{'}procS, \text{'}procT) \text{ Proc } \Rightarrow (\text{'}procS, \text{'}procT) \text{ Proc } \Rightarrow \text{ bool } (- \text{ R[\cdot]} - [75, 75] 80) \]

**Where**

\[ P \text{ R[\cdot]} Q \equiv (P, Q) \in \text{indRel} \]

**Lemma (in encoding) indRel-symm:**

**Shows** $\text{sym indRel}$

\[ \langle \text{proof} \rangle \]

**Inductive-set (in encoding) indRelEQ**

\[ \text{indRelEQ : } (((\text{'}procS, \text{'}procT) \text{ Proc}) \times ((\text{'}procS, \text{'}procT) \text{ Proc})) \text{ set} \]

**Where**

\begin{align*}
\text{encR} & : (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelEQ} | \\
\text{encL} & : (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{indRelEQ} | \\
\text{target} & : (\text{TargetTerm } T, \text{TargetTerm } T) \in \text{indRelEQ} | \\
\text{trans} & : [(P, Q) \in \text{indRelEQ}; (Q, R) \in \text{indRelEQ}] \Rightarrow (P, R) \in \text{indRelEQ}
\end{align*}

**Abbreviation (in encoding) indRelEQinfix**

\[ (\text{'}procS, \text{'}procT) \text{ Proc } \Rightarrow (\text{'}procS, \text{'}procT) \text{ Proc } \Rightarrow \text{ bool } (- \text{ ~[\cdot]} - [75, 75] 80) \]

**Where**

\[ P \text{ ~[\cdot]} Q \equiv (P, Q) \in \text{indRelEQ} \]

**Lemma (in encoding) indRelEQ-refl:**

**Shows** $\text{refl indRelEQ}$

\[ \langle \text{proof} \rangle \]

**Lemma (in encoding) indRelEQ-is-preorder:**

**Shows** $\text{preorder indRelEQ}$

32
lemma (in encoding) indRelEQ-symm:
  shows sym indRelEQ
  ⟨proof⟩

lemma (in encoding) indRelEQ-is-equivalence:
  shows equivalence indRelEQ
  ⟨proof⟩

lemma (in encoding) refl-trans-closure-of-indRel:
  shows indRelEQ = indRel"^+
  ⟨proof⟩

lemma (in encoding) refl-symm-trans-closure-of-indRel:
  shows indRelEQ = (symcl (indRel")) +
  ⟨proof⟩

lemma (in encoding) symm-closure-of-indRelR:
  shows indRel = symcl indRelR
  and indRelEQ = (symcl (indRel")) +
  ⟨proof⟩

lemma (in encoding) symm-closure-of-indRelL:
  shows indRel = symcl indRelL
  and indRelEQ = (symcl (indRel")) +
  ⟨proof⟩

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⟩

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⟩

lemma (in encoding) indRel-cond-respection-iff-indRelR-cond-respection:
  fixes Pred :: (′procS, ′procT) Proc ⇒ bool
  shows (∀ 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⟩

lemma (in encoding) indRel-cond-binary-respection-iff-indRelR-cond-binary-respection:
  fixes Pred :: (′procS, ′procT) Proc ⇒ ′b ⇒ bool
  shows (∀ 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⟩

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
⟨proof⟩

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)
  ⟨proof⟩

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\[\|RT\rightarrow\]\ [- [75, 75, 75] 80])
  where
  P \[\|RT\rightarrow\] Q \equiv (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] \implies (P, R) ∈ indRelRTPO TRel

abbreviation (in encoding) indRelRTPOinfix
  :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
  (- R\[\|RT\rightarrow\]\ [- [75, 75, 75] 80])
  where
  P \[\|RT\rightarrow\] Q \equiv (P, Q) ∈ indRelRTPO TRel

lemma (in encoding) indRelRTPO-refl:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows refl (indRelRTPO TRel)
  ⟨proof⟩

lemma (in encoding) refl-trans-closure-of-indRelRT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows indRelRTPO TRel = (indRelRT TRel)∗
  ⟨proof⟩

lemma (in encoding) indRelRTPO-is-preorder:
**inductive-set**

**abbreviation**

**lemma** (in encoding) transitive-closure-of-TRel-to-indRelRTPO:

fixes $\text{TRel} :: ('\text{procT} \times '\text{procT})$ set
assumes $\text{reflT} :: \text{refl TRel}$
shows preorder $(\text{indRelRTPO TRel})$
(proof)

**lemma** (in encoding) indRelR-modulo-pred-impl-indRelRT-modulo-pred:

fixes $\text{Pred} :: (('\text{procS}', '\text{procT}) \text{ Proc} \times ('\text{procS}', '\text{procT}) \text{ Proc}) \Rightarrow \text{bool}$
shows $(\forall \text{Trel}. \forall (\text{TP}, \text{TQ}) \in \text{Trel}. \text{Pred} (\text{TargetTerm} \text{ TP}, \text{TargetTerm} \text{ TQ}) \leftarrow (\forall (\text{P}, \text{Q}) \in \text{indRelRT} \text{ TRel. Pred} (\text{P}, \text{Q})))$
(proof)

**lemma** (in encoding) indRelRT-iff-exists-source-target-relation:

fixes $\text{TRel} :: ('\text{procT} \times '\text{procT})$ set
and $\text{Pred} :: (('\text{procS}', '\text{procT}) \text{ Proc} \times ('\text{procS}', '\text{procT}) \text{ Proc}) \Rightarrow \text{bool}$
assumes $\text{reflCond} :: \forall \text{P. Pred} (\text{P}, \text{P})$
and $\text{transCond} :: \forall \text{P Q R. Pred} (\text{P}, \text{Q}) \wedge \text{Pred} (\text{Q}, \text{R}) \rightarrow \text{Pred} (\text{P}, \text{R})$
shows $(\forall (\text{P}, \text{Q}) \in \text{indRelRT} \text{ TRel. Pred} (\text{P}, \text{Q})) = (\forall (\text{P}, \text{Q}) \in \text{indRelRTPO TRel. Pred} (\text{P}, \text{Q}))$
(proof)

**lemma** (in encoding) indRelR-modulo-pred-impl-indRelRTPO-modulo-pred:

fixes $\text{TRel} :: ('\text{procT} \times '\text{procT})$ set
and $\text{Pred} :: (('\text{procS}', '\text{procT}) \text{ Proc} \times ('\text{procS}', '\text{procT}) \text{ Proc}) \Rightarrow \text{bool}$
assumes $\text{reflCond} :: \forall \text{P. Pred} (\text{P}, \text{P})$
and $\text{transCond} :: \forall \text{P Q R. Pred} (\text{P}, \text{Q}) \wedge \text{Pred} (\text{Q}, \text{R}) \rightarrow \text{Pred} (\text{P}, \text{R})$
shows $(\forall (\text{P}, \text{Q}) \in \text{indRelRT} \text{ TRel. Pred} (\text{P}, \text{Q})) = (\forall (\text{P}, \text{Q}) \in \text{indRelRTPO TRel. Pred} (\text{P}, \text{Q}))$
(proof)

The relation $\text{indRelRT}$ includes $\text{TRel}$ and relates literal translations and their source terms.

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

:: $(('\text{procT} \times '\text{procT})$ set $\Rightarrow ((('\text{procS}', '\text{procT}) \text{ Proc}) \times ((('\text{procS}', '\text{procT}) \text{ Proc}) \Rightarrow \text{bool})$)
for $\text{Trel} :: ('\text{procT} \times '\text{procT})$ set
where
$\text{encL} :: (\text{TargetTerm} (\text{[S]}), \text{SourceTerm} \text{ S} \in \text{indRelLT} \text{ TRel | target: (T1, T2)} \in \text{Trel} \Rightarrow (\text{TargetTerm} \text{ T1, TargetTerm} \text{ T2}) \in \text{indRelLT} \text{ TRel}$

**abbreviation** (in encoding) $\text{indRelLTinfix}$

:: $(('\text{procS}', '\text{procT}) \text{ Proc} \Rightarrow ('\text{procT} \times '\text{procT})$ set $\Rightarrow ('\text{procS}', '\text{procT}) \text{ Proc} \Rightarrow \text{bool})$)
where
$\text{P R}[\text{LT} <\text{TRel} > \text{Q} \equiv (\text{P}, \text{Q}) \in \text{indRelLT} \text{ TRel}$

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

:: $(('\text{procT} \times '\text{procT})$ set $\Rightarrow (((('\text{procS}', '\text{procT}) \text{ Proc}) \times (((('\text{procS}', '\text{procT}) \text{ Proc}) \Rightarrow \text{bool})$)
for TRel :: (’procT × ’procT) set
where
cencL: (TargetTerm ([S]), SourceTerm S) ∈ indRelLTPO TRel |
csource: (SourceTerm S, TargetTerm ([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
(- ≲[\text{LT}]<- - [75, 75, 75] 80)
where
P ≲[\text{LT}]<- P \equiv (P, Q) ∈ indRelLTPO TRel

lemma (in encoding) indRelLTPO-refl:
fixes TRel :: (’procT × ’procT) set
assumes refl: refl TRel
shows refl (indRelLTPO TRel)
(proof)

lemma (in encoding) refl-trans-closure-of-indRelLT:
fixes TRel :: (’procT × ’procT) set
assumes refl: refl TRel
shows indRelLTPO TRel = (indRelLT TRel)∗
(proof)

inductive-set (in encoding) indRelT
:: (’procT × ’procT) set ⇒ (((’procS, procT) Proc) × ((’procS, procT) Proc)) set
for TRel :: (’procT × ’procT) set
where
cencR: (SourceTerm S, TargetTerm ([S])) ∈ indRelT TRel |
cencL: (TargetTerm ([S]), SourceTerm S) ∈ indRelT TRel |
target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelT TRel

abbreviation (in encoding) indRelTinfix
:: (procS, procT) Proc ⇒ (’procT × ’procT) set ⇒ (procS, procT) Proc ⇒ bool
(- ⇒[\text{T}]<-> - [75, 75, 75] 80)
where
P ⇒[\text{T}]<-> P \equiv (P, Q) ∈ indRelT TRel

lemma (in encoding) indRelT-symm:
fixes TRel :: (’procT × ’procT) set
assumes symm: sym TRel
shows symm (indRelT TRel)
(proof)

inductive-set (in encoding) indRelTEQ
:: (’procT × ’procT) set ⇒ (((’procS, procT) Proc) × ((’procS, procT) Proc)) set
for TRel :: (’procT × ’procT) set
where
cencR: (SourceTerm S, TargetTerm ([S])) ∈ indRelTEQ TRel |
cencL: (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
(- ≲[\text{LT}]<-> - [75, 75, 75] 80)
where
P ≲[\text{LT}]<-> P \equiv (P, Q) ∈ indRelTEQ TRel

lemma (in encoding) indRelTEQ-refl:
\begin{verbatim}
fixes TRel :: ('procT × 'procT) set
assumes refl: refl TRel
shows refl (indRelTEQ TRel)
⟨proof⟩

lemma (in encoding) indRelTEQ-symm:
fixes TRel :: ('procT × 'procT) set
assumes symm: sym TRel
shows sym (indRelTEQ TRel)
⟨proof⟩

lemma (in encoding) refl-trans-closure-of-indRelT:
fixes TRel :: ('procT × 'procT) set
assumes refl: refl TRel
shows indRelTEQ TRel = (indRelT TRel)^*
⟨proof⟩

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⟩

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⟩

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⟩

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 × 'procT) set
and TP TQ :: 'procT
assumes rel: TargetTerm TP R[]RT<TRel> TargetTerm TQ
shows (TP, TQ) ∈ TRel
⟨proof⟩

lemma (in encoding) indRelLT-to-TRel:
fixes TRel :: ('procT × 'procT) set
and TP TQ :: 'procT
assumes rel: TargetTerm TP R[]LT<TRel> TargetTerm TQ
shows (TP, TQ) ∈ TRel
⟨proof⟩

lemma (in encoding) indRelT-to-TRel:
fixes TRel :: ('procT × 'procT) set
and TP TQ :: 'procT
assumes rel: TargetTerm TP R[]T<TRel> TargetTerm TQ

\end{verbatim}
shows \((TP, TQ) \in TRel\)

\(\langle proof\rangle\)

If the preorders \(\text{indRelRTPO}, \text{indRelLTPO},\) or the equivalence \(\text{indRelTEQ}\) contain a pair of terms, then the pair of target terms that is related to these two terms is also related by the reflexive and transitive closure of the considered target term relation.

**lemma (in encoding) \text{indRelRTPO-to-TRel}:**

**fixes** \(TRel::('procT \times 'procT) set\)

**and** \(P Q::('procS, 'procT) Proc\)

**assumes** \(rel::P \subseteq[1]RT<TRel\>

**shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q \longrightarrow SP = SQ\)

\(\forall SP TQ. SP \in S P \land TQ \in T Q\)

\(\longrightarrow ([SP], [TQ]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\forall TP SQ. TP \in T P \land SQ \in S Q\)

\(\longrightarrow (TP, [SQ]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\forall TP TQ. TP \in T P \land TQ \in T Q\)

\(\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\langle proof\rangle\)

**lemma (in encoding) \text{indRelLTPO-to-TRel}:**

**fixes** \(TRel::('procT \times 'procT) set\)

**and** \(P Q::('procS, 'procT) Proc\)

**assumes** \(rel::P \subseteq[1]LT<TRel\>

**shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q \longrightarrow SP = SQ\)

\(\forall SP TQ. SP \in S P \land TQ \in T Q\)

\(\longrightarrow (TP, [SQ]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\forall TP TQ. TP \in T P \land TQ \in T Q\)

\(\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\langle proof\rangle\)

**lemma (in encoding) \text{indRelTEQ-to-TRel}:**

**fixes** \(TRel::('procT \times 'procT) set\)

**and** \(P Q::('procS, 'procT) Proc\)

**assumes** \(rel::P \sim[\sim]T<TRel\>

**shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q\)

\(\longrightarrow ([SP], [SQ]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\forall SP TQ. SP \in S P \land TQ \in T Q\)

\(\longrightarrow (TP, [SQ]) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\forall TP TQ. TP \in T P \land TQ \in T Q\)

\(\longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

\(\langle proof\rangle\)

**lemma (in encoding) \text{trans-closure-of-TRel-refl-cond}:**

**fixes** \(TRel::('procT \times 'procT) set\)

**and** \(TP TQ::'procT\)

**assumes** \(TP, TQ \in (TRel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S])\}^+\)

**shows** \(TP, TQ \in TRel^+\)

\(\langle proof\rangle\)

Note that if \(\text{indRelRTPO}\) relates a source term \(S\) to a target term \(T\), then the translation of \(S\) is equal to \(T\) or \(\text{indRelRTPO}\) also relates the translation of \(S\) to \(T\).

**lemma (in encoding) \text{indRelRTPO-relates-source-target}:**

**fixes** \(TRel::('procT \times 'procT) set\)

**and** \(S::'procS\)

**and** \(T::'procT\)

**assumes** \(\text{pair::SourceTerm } S \subseteq[\subseteq]RT<TRel> \text{TargetTerm } T\)

**shows** \(\text{TargetTerm } ([S]), \text{TargetTerm } T \in (\text{indRelRTPO } TRel)^+\)

\(\langle proof\rangle\)

If \(\text{indRelRTPO}, \text{indRelLTPO},\) or \(\text{indRelTPO}\) preserves barbs then so does the corresponding target
lemma (in encoding-wrt-barbs) rel-with-target-impl-TRel-preserves-barbs:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
  assumes preservation: rel-preserves-barbs Rel (STCalWB SWB TWB)
  and targetInRel: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  shows rel-preserves-barbs TRel TWB
  ⟨proof⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

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⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

If indRelRTPO, indRelLTPO, or indRelTPO reflects barbs then so does the corresponding target term relation.
shows rel-reflects-barbs TRel TWB

(proof)

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
  (proof)

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
  (proof)

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
  (proof)

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)

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
  (proof)

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
  (proof)

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
  (proof)

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 respect: rel-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs TRel TWB
  (proof)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-respects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes respect: rel-respects-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs TRel TWB
  (proof)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-respects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes respect: rel-respects-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs TRel TWB
  (proof)

40
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
  ⟨proof⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

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
  ⟨proof⟩

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: ∀ 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⁺) Target
  ⟨proof⟩

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
  shows weak-reduction-simulation (TRel⁺) Target
  ⟨proof⟩

lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-reduction-simulation (indRelLTPO TRel) (STCal Source Target)
  shows weak-reduction-simulation (TRel⁺) Target
  ⟨proof⟩

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⟩

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

\(\langle \text{proof} \rangle\)

**Lemma (in encoding)** \(\text{indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{assumes sim: weak-reduction-simulation ((indRelLTPO TRel)^{-1}) (STCal Source Target)}\\
\text{shows weak-reduction-simulation ((TRel^+)^{-1}) Target}\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{and Rel :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \text{ set}}\\
\text{assumes sim: weak-reduction-simulation Rel (STCal Source Target)}\\
\text{and target: \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in Rel}\\
\text{and trel: \forall T1 T2. (\text{TargetTerm T1}, \text{TargetTerm T2}) \in Rel \rightarrow (T1, T2) \in TRel^+}\\
\text{shows weak-reduction-simulation (TRel^+)} Target\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{indRelTEQ-impl-TRel-is-weak-reduction-simulation}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{assumes sim: weak-reduction-simulation (\text{indRelTEQ TRel}) (STCal Source Target)}\\
\text{shows weak-reduction-simulation (TRel^+)} Target\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{rel-with-target-impl-transC-TRel-is-weak-reduction-simulation}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{and Rel :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \text{ set}}\\
\text{assumes sim: strong-reduction-simulation Rel (STCal Source Target)}\\
\text{and target: \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in Rel}\\
\text{and trel: \forall T1 T2. (\text{TargetTerm T1}, \text{TargetTerm T2}) \in Rel \rightarrow (T1, T2) \in TRel^+}\\
\text{shows strong-reduction-simulation (TRel^+)} Target\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{indRelRTPO-impl-TRel-is-weak-reduction-simulation}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{assumes sim: strong-reduction-simulation (\text{indRelRTPO TRel}) (STCal Source Target)}\\
\text{shows strong-reduction-simulation (TRel^+)} Target\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{indRelLTPO-impl-TRel-is-weak-reduction-simulation}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{assumes sim: strong-reduction-simulation (\text{indRelLTPO TRel}) (STCal Source Target)}\\
\text{shows strong-reduction-simulation (TRel^+)} Target\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev}\\)

\text{fixes} \ TRel :: (\text{procT} \times \text{procT}) \text{ set}\\
\text{assumes sim: strong-reduction-simulation ((\text{indRelRTPO TRel})^{-1}) (STCal Source Target)}\\
\text{shows strong-reduction-simulation ((TRel^+)^{-1}) Target}\\
\text{\(\langle \text{proof} \rangle\)}

**Lemma (in encoding)** \(\text{indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev}\\)
lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-reduction-simulation ((indRelLTPO TRel)^-1) (STCal Source Target)
  shows strong-reduction-simulation ((TRel^+)^-1) Target
  (proof)

lemma (in encoding) indRelTEQ-impl-TRel-is-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-reduction-simulation (indRelTEQ TRel) (STCal Source Target)
  shows strong-reduction-simulation (TRel^+) Target
  (proof)

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-barbed-simulation (indRelRTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-simulation (TRel^+) TWB
  (proof)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-barbed-simulation (indRelLTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-simulation (TRel^+) TWB
  (proof)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-barbed-simulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows weak-barbed-simulation (TRel^+) TWB
  (proof)

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-barbed-simulation (indRelRTPO TRel) (STCalWB SWB TWB)
  shows strong-barbed-simulation (TRel^+) TWB
  (proof)

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)

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)

lemma (in encoding-wrt-barbs) rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation:

If indRelRTPO, indRelLTPO, or indRelTEQ is a contrasimulation then so is the corresponding target
term relation.

lemma (in encoding) rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation:
fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-contrasimulation \ Rel (STCal Source Target)}$
\[\text{and}\] target: $\forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel$
\[\text{and}\ trel: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \rightarrow (T1, T2) \in TRel^*$

shows weak-reduction-contrasimulation ($TRel^*$) Target

\[\text{(proof)}\]

lemma (in encoding) $\text{indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-contrasimulation (indRelRTPO TRel) (STCal Source Target)}$

shows weak-reduction-contrasimulation ($TRel^*$) Target

\[\text{(proof)}\]

lemma (in encoding) $\text{indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-contrasimulation (indRelLTPO TRel) (STCal Source Target)}$

shows weak-reduction-contrasimulation ($TRel^*$) Target

\[\text{(proof)}\]

lemma (in encoding) $\text{rel-with-target-impl-refC-transC-TRel-is-weak-reduction-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-contrasimulation \ Rel (STCal Source Target)}$
\[\text{and}\ target: \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel$
\[\text{and}\ trel: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \rightarrow (T1, T2) \in TRel^*$

shows weak-reduction-contrasimulation ($TRel^*$) Target

\[\text{(proof)}\]

lemma (in encoding) $\text{indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-contrasimulation (indRelTEQ TRel) (STCal Source Target)}$

shows weak-reduction-contrasimulation ($TRel^*$) Target

\[\text{(proof)}\]

lemma (in encoding-wrt-barbs) $\text{indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-barbed-contrasimulation (indRelRTPO TRel) (STCalWB SWB TWB)}$

shows weak-barbed-contrasimulation ($TRel^*$) TWB

\[\text{(proof)}\]

lemma (in encoding-wrt-barbs) $\text{indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-barbed-contrasimulation (indRelLTPO TRel) (STCalWB SWB TWB)}$

shows weak-barbed-contrasimulation ($TRel^*$) TWB

\[\text{(proof)}\]

lemma (in encoding-wrt-barbs) $\text{indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-barbed-contrasimulation (indRelTEQ TRel) (STCalWB SWB TWB)}$

shows weak-barbed-contrasimulation ($TRel^*$) TWB

\[\text{(proof)}\]

If indRelRTPO, indRelLTPO, or indRelTEQ is a coupled simulation then so is the corresponding target term relation.

lemma (in encoding) $\text{indRelRTPO-impl-TRel-is-weak-reduction-coupled-simulation:}$

fixes $TRel : (('procT \times 'procT) set$
\[\text{and}\] $Rel : (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set$

assumes $\text{conSim: weak-reduction-coupled-simulation (indRelRTPO TRel) (STCal Source Target)}$

shows weak-reduction-coupled-simulation ($TRel^*$) Target

\[\text{(proof)}\]
lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation:
  fixes TRel :: (’procT × ’procT) set
  assumes couSim: weak-reduction-coupled-simulation (indRelLTPO TRel) (STCal Source Target)
  shows weak-reduction-coupled-simulation (TRel') Target
    ⟨proof⟩

lemma (in encoding) indRelTEQ-impl-TRel-is-weak-reduction-coupled-simulation:
  fixes TRel :: (’procT × ’procT) set
  assumes couSim: weak-reduction-coupled-simulation (indRelTEQ TRel) (STCal Source Target)
  shows weak-reduction-coupled-simulation (TRel') Target
    ⟨proof⟩

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-coupled-simulation:
  fixes TRel :: (’procT × ’procT) set
  assumes couSim: weak-barbed-coupled-simulation (indRelRTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-coupled-simulation (TRel') TWB
    ⟨proof⟩

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-coupled-simulation:
  fixes TRel :: (’procT × ’procT) set
  assumes couSim: weak-barbed-coupled-simulation (indRelLTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-coupled-simulation (TRel') TWB
    ⟨proof⟩

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-coupled-simulation:
  fixes TRel :: (’procT × ’procT) set
  assumes couSim: weak-barbed-coupled-simulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows weak-barbed-coupled-simulation (TRel') TWB
    ⟨proof⟩

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 × ’procT) set
    and Rel :: ((’procS, ’procT) Proc × (’procS, ’procT) Proc) set
  assumes corSim: weak-reduction-correspondence-simulation 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-correspondence-simulation (TRel') Target
    ⟨proof⟩

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
    ⟨proof⟩

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
    ⟨proof⟩

lemma (in encoding) rel-with-target-impl-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: ∀ T1 T2. (T1, T2) ∈ TRel −→ (TargetTerm T1, TargetTerm T2) ∈ Rel
and \( trel: \forall T1, T2. (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+ \)

shows weak-reduction-correspondence-simulation (TRel+) Target
(proof)

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+) TWB
(proof)

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)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-correspondence-simulation:

fixes TRel :: ('procT × 'procT) set

assumes corSim: weak-barbed-correspondence-simulation (indRelLTPO TRel) (STCalWB SWB TWB)

shows weak-barbed-correspondence-simulation (TRel+) TWB
(proof)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-correspondence-simulation:

fixes TRel :: ('procT × 'procT) set

assumes corSim: weak-barbed-correspondence-simulation (indRelTEQ TRel) (STCalWB SWB TWB)

shows weak-barbed-correspondence-simulation (TRel+) TWB
(proof)

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 × 'procT) set

and Rel :: ('(procS, 'procT) Proc) × ('(procS, 'procT) Proc) set

assumes bisim: weak-reduction-bisimulation Rel (STCal Source Target)

and target: \( \forall T1, T2. (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel \)

and trel: \( \forall T1, T2. (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+ \)

shows weak-reduction-bisimulation (TRel+) Target
(proof)

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-bisimulation:

fixes TRel :: ('procT × 'procT) set

assumes bisim: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

shows weak-reduction-bisimulation (TRel+) Target
(proof)

lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-bisimulation:

fixes TRel :: ('procT × 'procT) set

assumes bisim: weak-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)

shows weak-reduction-bisimulation (TRel+) Target
(proof)

lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation:

fixes TRel :: ('procT × 'procT) set

and Rel :: ('(procS, 'procT) Proc) × ('(procS, 'procT) Proc) set

assumes bisim: weak-reduction-bisimulation Rel (STCal Source Target)

and target: \( \forall T1, T2. (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel \)

and trel: \( \forall T1, T2. (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+ \)

shows weak-reduction-bisimulation (TRel+) Target
(proof)
lemma (in encoding) indRelTEQ-impl-TRel-is-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
  shows weak-reduction-bisimulation (TRel+) Target
  (proof)

lemma (in encoding) rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes bisim: strong-reduction-bisimulation 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 strong-reduction-bisimulation (TRel+) Target
  (proof)

lemma (in encoding) indRelRTPO-impl-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
  shows strong-reduction-bisimulation (TRel+) Target
  (proof)

lemma (in encoding) indRelLTPO-impl-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: strong-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)
  shows strong-reduction-bisimulation (TRel+) Target
  (proof)

lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes bisim: strong-reduction-bisimulation 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 strong-reduction-bisimulation (TRel+) Target
  (proof)

lemma (in encoding) indRelTEQ-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: weak-barbed-bisimulation (indRelTEQ TRel) (STCal WB SWB TBW)
  shows weak-barbed-bisimulation (TRel+) TBW
  (proof)

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 TBW)
  shows weak-barbed-bisimulation (TRel+) TBW
  (proof)

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 TBW)
  shows weak-barbed-bisimulation (TRel+) TBW
  (proof)

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 TBW)
  shows weak-barbed-bisimulation (TRel+) TBW
  (proof)
lemma (in encoding-wrt-barbs) indRelLTPO-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)

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)

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 × procS) (procT × procT) (procS × procS) (procT × procT)
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, 80]
where
P R[1]\R<5SRel,TRel> Q ≡ (P, Q) ∈ indRelRST SRel TRel

inductive-set (in encoding) indRelRSTPO
:: (procS × procS) set ⇒ (procT × procT) set
⇒ (procS × procS) (procT × procT) (procS × procS) (procT × procT)
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[, ] R<, > [75, 75, 75, 80]
where
P R[1]\R<5SRel,TRel> Q ≡ (P, Q) ∈ indRelRSTPO SRel TRel

lemma (in encoding) indRelRSTPO-refl:
fixes SRel :: (procS × procS) set
and TRel :: (procT × procT) set

48
lemma (in encoding) indRelRSTPO-trans:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  shows trans (indRelRSTPO SRel TRel)
  (proof)

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)

inductive-set (in encoding) indRelLST
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) 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<.-,>-] [75, 75, 75, 75] 80 L<SRel,TRel> Q ≡ (P, Q) ∈ indRelLST SRel TRel

inductive-set (in encoding) indRelLSTPO
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) 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 (- \V [L<.-,>-] [75, 75, 75, 75] 80)
  where
    P \V [L<.-,>-] [75, 75, 75, 75] 80 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)
  (proof)
lemma \(\text{in encoding}\) \(\text{indRelLSTPO-trans:}\)
\[
\text{fixes } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{and } TRel : (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{shows } \text{trans} \ (\text{\textquotesingle}\text{indRelLSTPO \ SRel \ TRel}\text{\textquotesingle})
\]
\(\text{(proof)}\)

lemma \(\text{in encoding}\) \(\text{refl-trans-closure-of-indRelLST:}\)
\[
\text{fixes } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{and } TRel : (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{assumes } \text{reflS : refl SRel}
\text{and } \text{reflT : refl TRel}
\text{shows } \text{indRelLSTPO \ SRel \ TRel} = (\text{indRelLST \ SRel \ TRel})^*
\]
\(\text{(proof)}\)

inductive-set \(\text{in encoding}\) \(\text{indRelLST}\)
\[
:: (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{for } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{and } TRel : (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{where}
\]
\[
\text{encR : (SourceTerm S, TargetTerm ([S]))} \in \text{indRelLST \ SRel \ TRel} | \text{encL: (TargetTerm ([S]), SourceTerm S)} \in \text{indRelLST \ SRel \ TRel} | \text{source: (S1, S2) } \in \text{SRel} \rightarrow (\text{SourceTerm S1, SourceTerm S2}) \in \text{indRelLST \ SRel \ TRel} | \text{target: (T1, T2) } \in \text{TRel} \rightarrow (\text{TargetTerm T1, TargetTerm T2}) \in \text{indRelLST \ SRel \ TRel}
\]

abbreviation \(\text{in encoding}\) \(\text{indRelLSTinfix}\)
\[
:: (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{where}
\]
\[
P \rightarrow [50 \leftarrow [75, 75, 75, 80]} \set
\]

lemma \(\text{in encoding}\) \(\text{indRelLST-symm:}\)
\[
\text{fixes } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{and } TRel : (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{assumes } \text{symmS : sym SRel}
\text{and } \text{symmT : sym TRel}
\text{shows } \text{sym} \ (\text{\textquotesingle}\text{indRelLST \ SRel \ TRel}\text{\textquotesingle})
\]
\(\text{(proof)}\)

inductive-set \(\text{in encoding}\) \(\text{indRelSTEQ}\)
\[
:: (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{for } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{and } TRel : (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{where}
\]
\[
\text{encR : (SourceTerm S, TargetTerm ([S]))} \in \text{indRelSTEQ \ SRel \ TRel} | \text{encL : (TargetTerm ([S]), SourceTerm S)} \in \text{indRelSTEQ \ SRel \ TRel} | \text{source: (S1, S2) } \in \text{SRel} \rightarrow (\text{SourceTerm S1, SourceTerm S2}) \in \text{indRelSTEQ \ SRel \ TRel} | \text{target: (T1, T2) } \in \text{TRel} \rightarrow (\text{TargetTerm T1, TargetTerm T2}) \in \text{indRelSTEQ \ SRel \ TRel} | \text{trans: [\text{\textquotesingle}P, Q\text{\textquotesingle}] \in \text{indRelSTEQ \ SRel \ TRel}; (\text{\textquotesingle}Q, R\text{\textquotesingle}) \in \text{indRelSTEQ \ SRel \ TRel}]} \rightarrow (\text{\textquotesingle}P, R\text{\textquotesingle}) \in \text{indRelSTEQ \ SRel \ TRel}
\]

abbreviation \(\text{in encoding}\) \(\text{indRelSTEQinfix}\)
\[
:: (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\text{where}
\]
\[
P \rightarrow [50 \leftarrow [75, 75, 75, 80]} \set
\]

lemma \(\text{in encoding}\) \(\text{indRelSTEQ-refl:}\)
\[
\text{fixes } SRel : (\text{\textquotesingle}\text{procS} \times \text{procS}\text{\textquotesingle}) \rightarrow (\text{\textquotesingle}\text{procT} \times \text{procT}\text{\textquotesingle}) \set
\]
assumes reflT: refl TRel
shows refl (indRelSTEQ SRel TRel)
⟨proof⟩

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)
⟨proof⟩

lemma (in encoding) indRelSTEQ-trans:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
shows trans (indRelSTEQ SRel TRel)
⟨proof⟩

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⟩

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⟩

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⟩

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⟩

lemma (in encoding) symm-trans-closure-of-indRelRSTPO:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes symmS: sym SRel
and symmT: sym TRel
shows indRelSTEQ SRel TRel = (symcl (indRelRSTPO SRel TRel))^+
⟨proof⟩
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⟩
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
⟨proof⟩
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
⟨proof⟩
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
⟨proof⟩
lemma (in encoding) indRelLST-to-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  assumes rel: TargetTerm TP R ]L<,SRel,TRel> TargetTerm TQ
  shows (TP, TQ) ∈ TRel
⟨proof⟩
lemma (in encoding) indRelST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP R ]<,SRel,TRel> SourceTerm SQ
  shows (SP, SQ) ∈ SRel
⟨proof⟩
lemma (in encoding) indRelST-to-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  assumes rel: TargetTerm TP R ]<,SRel,TRel> TargetTerm TQ
  shows (TP, TQ) ∈ TRel
⟨proof⟩
If the relations indRelRSTPO or indRelLSTPO contain a pair of target terms, then this pair is also
related by the transitive closure of the considered target term relation. Similarly a pair of source terms is related by the transitive closure of the source term relation. A pair of a source and a target term results from the combination of pairs in the source relation, the target relation, and the encoding function. Note that, because of the symmetry, no similar condition holds for indRelSTEQ.

**Lemma (in encoding) indRelRSTPO-to-SRel-and-TRel:**

- **Proof:**
  
  - `fixes SRel :: (procS × procS) set`
  - `and TRel :: (procT × procT) set`
  - `and P Q :: (procS, procT) Proc`
  - `assumes P ≤[ ] R < SRel, TRel > Q`
  - `shows ∀ SP SQ. SP ∈ S P ∧ SQ ∈ S Q → (SP, SQ) ∈ SRel+`
  - `and ∀ TP SQ. TP ∈ T P ∧ SQ ∈ S Q → false`
  - `and ∀ TP TQ. TP ∈ T P ∧ TQ ∈ T Q → (TP, TQ) ∈ TRel+`

**Lemma (in encoding) indRelLSTPO-to-SRel-and-TRel:**

- **Proof:**
  
  - `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 ∀ TP SQ. TP ∈ T P ∧ SQ ∈ S Q → false`
  - `and ∀ TP TQ. TP ∈ T P ∧ TQ ∈ T Q → (TP, TQ) ∈ TRel+`

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:**

- **Proof:**
  
  - `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`

**Lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-preserve-barbs:**

- **Proof:**
  
  - `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`

**Lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-preserve-barbs:**

- **Proof:**
  
  - `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`

**Lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-preserve-barbs:**

- **Proof:**
  
  - `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`
lemma (in encoding-wrt-barbs) rel-with-source-impl-SSSRel-weakly-preserves-barbs:
  fixes SRel :: (‘procS × ‘procS) 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)

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SSSRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: (‘procS × ‘procS) set
  and TRel :: (‘procT × ‘procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelRSTPO SRel TRel) SRel SWB
  shows rel-weakly-preserves-barbs TRel TWB
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SSSRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: (‘procS × ‘procS) set
  and TRel :: (‘procT × ‘procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelLSTPO SRel TRel) SRel SWB
  shows rel-weakly-preserves-barbs TRel TWB
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SSSRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: (‘procS × ‘procS) set
  and TRel :: (‘procT × ‘procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelSTEQ SRel TRel) SRel SWB
  shows rel-weakly-preserves-barbs TRel TWB
(proof)

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-SSSRel-reflects-barbs:
  fixes SRel :: (‘procS × ‘procS) 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)

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SSSRel-and-TRel-reflect-barbs:
  fixes SRel :: (‘procS × ‘procS) set
  and TRel :: (‘procT × ‘procT) set
  assumes reflection: rel-reflects-barbs (indRelRSTPO SRel TRel) SRel SWB
  shows rel-reflects-barbs TRel TWB
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SSSRel-and-TRel-reflect-barbs:
  fixes SRel :: (‘procS × ‘procS) set
  and TRel :: (‘procT × ‘procT) set
  assumes reflection: rel-reflects-barbs (indRelLSTPO SRel TRel) SRel SWB
  shows rel-reflects-barbs TRel TWB
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SSSRel-and-TRel-reflect-barbs:
  fixes SRel :: (‘procS × ‘procS) set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes reflection: rel-reflects-barbs $(\text{indRelSTEQ SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-reflects-barbs $S_{rel}$ $S_{WB}$
and rel-reflects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) rel-with-source-impl-Srel-weakly-reflects-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $R_{rel} : ((\text{'proc}_S, \text{'proc}_T) \text{Proc} \times (\text{'proc}_S, \text{'proc}_T) \text{Proc})$ set
assumes reflection: rel-weakly-reflects-barbs $R_{rel}$ $(\text{STCalWB SWB TWB})$
and sourceInRel: $\forall S_1 S_2. (S_1, S_2) \in S_{rel} \rightarrow (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in R_{rel}$
sows rel-weakly-reflects-barbs $S_{rel}$ $S_{WB}$
and rel-weakly-reflects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes reflection: rel-weakly-reflects-barbs $(\text{indRelRSTPO SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-weakly-reflects-barbs $S_{rel}$ $S_{WB}$
and rel-weakly-reflects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes reflection: rel-weakly-reflects-barbs $(\text{indRelLSTPO SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-weakly-reflects-barbs $S_{rel}$ $S_{WB}$
and rel-weakly-reflects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes reflection: rel-weakly-reflects-barbs $(\text{indRelSTEQ SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-weakly-reflects-barbs $S_{rel}$ $S_{WB}$
and rel-weakly-reflects-barbs $T_{rel}$ $T_{WB}$
(proof)

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 $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes respection: rel-respects-barbs $(\text{indRelRSTPO SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-respects-barbs $S_{rel}$ $S_{WB}$
and rel-respects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-respect-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes respection: rel-respects-barbs $(\text{indRelLSTPO SRel TRel})$ $(\text{STCalWB SWB TWB})$
sows rel-respects-barbs $S_{rel}$ $S_{WB}$
and rel-respects-barbs $T_{rel}$ $T_{WB}$
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-respect-barbs:
fixes $S_{rel} : (\text{'proc}_S \times \text{'proc}_S)$ set
and $T_{rel} : (\text{'proc}_T \times \text{'proc}_T)$ set
assumes respection: rel-respects-barbs $(\text{indRelSTEQ SRel TRel})$ $(\text{STCalWB SWB TWB})$
shows rel-respects-barbs SRel SWB
  and rel-respects-barbs TRel TWB

(\textit{proof})

\textbf{lemma (in encoding-wrt-barbs)} indRelRSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{procT}) set
  \textbf{assumes} respection: rel-weakly-respects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  \textbf{shows} rel-weakly-respects-barbs SRel SWB
  and rel-weakly-respects-barbs TRel TWB

(\textit{proof})

\textbf{lemma (in encoding-wrt-barbs)} indRelLSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{procT}) set
  \textbf{assumes} respection: rel-weakly-respects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  \textbf{shows} rel-weakly-respects-barbs SRel SWB
  and rel-weakly-respects-barbs TRel TWB

(\textit{proof})

\textbf{lemma (in encoding-wrt-barbs)} indRelSTEQ-impl-SRel-and-TRel-weakly-respect-barbs:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{procT}) set
  \textbf{assumes} respection: rel-weakly-respects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  \textbf{shows} rel-weakly-respects-barbs SRel SWB
  and rel-weakly-respects-barbs TRel TWB

(\textit{proof})

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.

\textbf{lemma (in encoding)} indRelRTPO-to-indRelTEQ:
  \textbf{fixes} TRel :: (\text{procT} \times \text{procT}) set
  and P Q :: (\text{procS}, \text{procT}) \text{Proc}
  \textbf{assumes} rel: P \preceq_{\text{\scriptsize RT}} T<\text{TRel}> Q
  and reflT: refl TRel
  \textbf{shows} P \sim_{\text{\scriptsize T\textless\textgreater TRel}} Q

(\textit{proof})

\textbf{lemma (in encoding)} indRelRTPO-to-indRelRSTPO:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{procT}) set
  and P Q :: (\text{procS}, \text{procT}) \text{Proc}
  \textbf{assumes} rel: P \preceq_{\text{\scriptsize RT}} T<\text{TRel}> Q
  and reflS: refl SRel
  \textbf{shows} P \preceq_{\text{\scriptsize SRel,TRel}} Q

(\textit{proof})

\textbf{lemma (in encoding)} indRelRSTPO-to-indRelSTEQ:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{procT}) set
  and P Q :: (\text{procS}, \text{procT}) \text{Proc}
  \textbf{assumes} rel: P \preceq_{\text{\scriptsize R}} T<R,SRel,TRel> Q
  \textbf{shows} P \sim_{\text{\scriptsize R} R<SRel,TRel> Q

(\textit{proof})

If indRelRTPO is a bisimulation and SRel is a reflexive bisimulation then also indRelRSTPO is a bisimulation.

\textbf{lemma (in encoding)} indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation:
  \textbf{fixes} SRel :: (\text{procS} \times \text{procS}) set
  and TRel :: (\text{procT} \times \text{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⟩
end

theory SuccessSensitivity
  imports SourceTargetRelation
begin

6 Success Sensitiveness and Barbs

To compare the abstract behavior of two terms, often some notion of success or successful termination is used. Daniele Gorla assumes a constant process (similar to the empty process) that represents successful termination in order to compare the behavior of source terms with their literal translations. Then an encoding is success sensitive if, for all source terms S, S reaches success iff the translation of S reaches success. Successful termination can be considered as some special kind of barb. Accordingly we generalize successful termination to the respectation of an arbitrary subset of barbs. An encoding respects a set of barbs if, for every source term S and all considered barbs a, S reaches a iff the translation of S reaches a.

abbreviation (in encoding-wrt-barbs) enc-weakly-preserves-barb-set :: 'barbs set ⇒ bool where
eenc-weakly-preserves-barb-set Barbs ≡ enc-preserves-binary-pred (λP a. a ∈ Barbs ∧ P ▼.a)

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

lemma (in encoding-wrt-barbs) enc-weakly-preserves-barbs-and-barb-set:
  shows enc-weakly-preserves-barbs = (∀ Barbs. enc-weakly-preserves-barb-set Barbs)
⟨proof⟩

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

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

lemma (in encoding-wrt-barbs) enc-weakly-reflects-barbs-and-barb-set:
  shows enc-weakly-reflects-barbs = (∀ Barbs. enc-weakly-reflects-barb-set Barbs)
⟨proof⟩

abbreviation (in encoding-wrt-barbs) enc-weakly-respects-barb-set :: 'barbs set ⇒ bool where

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

lemma (in encoding-wrt-barbs) enc-weakly-respects-barbs-and-barb-set:
  shows enc-weakly-respects-barbs = (∀ Barbs. enc-weakly-respects-barb-set Barbs)
⟨proof⟩

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
eenc-preserves-barb-set Barbs ≡ enc-preserves-binary-pred (λP a. a ∈ Barbs ∧ P ▼.a)

abbreviation (in encoding-wrt-barbs) enc-preserves-barbs :: bool where

57
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)

(proof)

**abbreviation (in encoding-wrt-barbs)** enc-preserves-binary-pred:

assoc (λP a. P↓a)

**abbreviation (in encoding-wrt-barbs)** enc-preserves-barb-set:

assoc (λP a. P↓a)

**lemma (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs:

shows enc-preserves-barb-set Barbs = (∀ Barbs. enc-preserves-barb-set Barbs)

(proof)

**abbreviation (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs:

assoc (λP a. P↓a)

**lemma (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs:

shows enc-preserves-barb-set Barbs = (∀ Barbs. enc-preserves-barb-set Barbs)

(proof)

**abbreviation (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs:

assoc (λP a. P↓a)

**lemma (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs:

shows enc-preserves-barb-set Barbs = (∀ Barbs. enc-preserves-barb-set Barbs)

(proof)

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-barbs Rel (STCalWB SWB TWB) Barbs)

(proof)

**lemma (in encoding-wrt-barbs)** enc-weakly-preserves-barbs-iff-source-target-rel:

shows enc-weakly-preserves-barbs-barb-set-barbs = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ rel-weakly-preserves-barbs-barb-set-barbs Rel (STCalWB SWB TWB))

(proof)

**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-barbs Rel (STCalWB SWB TWB) Barbs)

(proof)

**lemma (in encoding-wrt-barbs)** enc-preserves-barb-set-barbs-

shows enc-preserves-barbs-barb-set-barbs = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ rel-preserves-barbs-barb-set-barbs Rel (STCalWB SWB TWB))

(proof)

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 (reachabil-
ity/)existence of barbs.

**Lemma (in encoding-wrt-barbs) enc-weakly-reflects-barb-set-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-weakly-reflects-barb-set-barbs

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-weakly-reflects-barb-set} \text{Rel (STCalWB SWB TWB) Barbs})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-weakly-reflects-barbs-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-weakly-reflects-barbs

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-weakly-reflects-barbs} \text{Rel (STCalWB SWB TWB)})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-reflects-barb-set-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-reflects-barb-set

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-reflects-barb-set} \text{Rel (STCalWB SWB TWB) Barbs})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-reflects-barbs-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-reflects-barbs

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-reflects-barbs} \text{Rel (STCalWB SWB TWB)})\)

- **Proof**

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

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-weakly-respects-barb-set} \text{Rel (STCalWB SWB TWB) Barbs})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-weakly-respects-barbs-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-weakly-respects-barbs

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-weakly-respects-barbs} \text{Rel (STCalWB SWB TWB)})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-respects-barb-set-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-respects-barb-set

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-respects-barb-set} \text{Rel (STCalWB SWB TWB) Barbs})\)

- **Proof**

**Lemma (in encoding-wrt-barbs) enc-respects-barbs-iff-source-target-rel:**

- **Fixes** Barbs :: 'barbs set

- **Shows** enc-respects-barbs

  - \((\exists \text{Rel}. (\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S]) \in \text{Rel}) \land \text{rel-respects-barbs} \text{Rel (STCalWB SWB TWB)})\)

- **Proof**

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-weekly-respects-barb-set {success} = (\forall S. S \downarrow< SWB > success \leftrightarrow [S] \downarrow< TWB > success)
  (proof)

lemma (in encoding-wrt-barbs) success-sensitive-if-source-target-rel-weekly-respects-success:
  fixes success :: 'barbs
  shows enc-weekly-respects-barb-set {success}
  = (\exists Rel. (\forall S. (\SourceTerm S, \TargetTerm ([S])) \in Rel)
      \land rel-weekly-respects-barb-set Rel (STCalWB SWB TWB) {success})
  (proof)

lemma (in encoding-wrt-barbs) success-sensitive-if-source-target-rel-respects-success:
  fixes success :: 'barbs
  shows enc-weekly-respects-barb-set {success}
  = (\exists Rel. (\forall S. (\SourceTerm S, \TargetTerm ([S])) \in Rel)
      \land rel-weekly-respects-barb-set Rel (STCalWB SWB TWB) {success})
  (proof)

end

theory DivergenceReflection
  imports SourceTargetRelation
begin

7 Divergence Reflection

Divergence reflection forbids for encodings that introduce loops of internal actions. Thus they determine the practicability of encodings in particular with respect to implementations. An encoding reflects divergence if each loop in a target term result from the translation of a divergent source term.

abbreviation (in encoding) enc-preserves-divergence :: bool where
  enc-preserves-divergence \equiv enc-preserves-pred (\lambda P. P \mapsto ST\omega)

lemma (in encoding) divergence-preservation-cond:
  shows enc-preserves-divergence = (\forall S. S \mapsto (\Source)\omega \mapsto [S] \mapsto (\Target)\omega)
  (proof)

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

lemma (in encoding) divergence-reflection-cond:
  shows enc-reflects-divergence = (\forall S. [S] \mapsto (\Target)\omega \mapsto S \mapsto (\Source)\omega)
  (proof)

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

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

Apart from divergence reflection we consider divergence respection. An encoding respects divergence if each divergent source term is translated into a divergent target term and each divergent target term result from the translation of a divergent source term.

abbreviation (in encoding) enc-respects-divergence :: bool where
  enc-respects-divergence \equiv enc-respects-pred (\lambda P. P \mapsto ST\omega)
lemma (in encoding) divergence-respection-cond:
shows enc-respects-divergence = (\forall S. [S] \longrightarrow (Target)\omega \leftrightarrow S \longrightarrow (Source)\omega)
⟨proof⟩

abbreviation rel-respects-divergence
:: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
where
rel-respects-divergence Rel Cal ≡ rel-respects-pred Rel (λP. P \longrightarrow (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 = (∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-preserves-divergence Rel (STCal Source Target))
⟨proof⟩

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))
⟨proof⟩

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))
⟨proof⟩

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))
⟨proof⟩

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⟩

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 × 'procT) set ⇒ bool where**

\[
\text{operational-complete TRel} \equiv \\
\forall S S'. S \mapsto \text{Source} S' \rightarrow (\exists T. [S] \mapsto \text{Target} T \land ([S'], T) \in \text{TRel})
\]

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 × 'procT) set ⇒ bool where**

\[
\text{strongly-operational-complete TRel} \equiv \\
\forall S S'. S \mapsto \text{Source} S' \rightarrow (\exists T. [S] \mapsto \text{Target} T \land ([S'], T) \in \text{TRel})
\]

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 × 'procT) set ⇒ bool where**

\[
\text{weakly-operational-sound TRel} \equiv \\
\forall S T. [S] \mapsto \text{Target} T \rightarrow (\exists S' T'. S \mapsto \text{Source} S' \land T \mapsto \text{Target} T' \land ([S'], T) \in \text{TRel})
\]

And encoding is operational sound modulo TRel if each sequence of target term steps is the translation of a sequence of source term steps such that the derivatives are related by TRel. This criterion does not allow for intermediate states, i.e., does not allow to reach target term from an encoded source term that is not related by TRel to the translation of a source term.

**abbreviation (in encoding) operational-sound :: ('procT × 'procT) set ⇒ bool where**

\[
\text{operational-sound TRel} \equiv \forall S T. [S] \mapsto \text{Target} T \rightarrow (\exists S' T'. S \mapsto \text{Source} S' \land ([S'], T) \in \text{TRel})
\]

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 × 'procT) set ⇒ bool where**

\[
\text{strongly-operational-sound TRel} \equiv \\
\forall S T. [S] \mapsto \text{Target} T \rightarrow (\exists S' T'. S \mapsto \text{Source} S' \land ([S'], T) \in \text{TRel})
\]

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

\[
\text{weakly-operational-corresponding TRel} \equiv \\
\text{operational-complete TRel} \land \text{weakly-operational-sound TRel}
\]

Operational correspondence modulo is the combination of operational completeness and operational soundness modulo TRel.

**abbreviation (in encoding) operational-corresponding :: ('procT × 'procT) set ⇒ bool where**

\[
\text{operational-corresponding TRel} \equiv \text{operational-complete TRel} \land \text{operational-sound TRel}
\]

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

\[
\text{strongly-operational-corresponding TRel} \equiv \\
\text{strongly-operational-complete TRel} \land \text{strongly-operational-sound TRel}
\]
8.1 Trivial Operational Correspondence Results

Every encoding is (weakly) operational corresponding modulo the all relation on target terms.

**Lemma (in encoding) operational-correspondencemodulo-all-relation:**

- shows operational-complete \{ (T1, T2). True \}
- and weakly-operational-sound \{ (T1, T2). True \}
- and operational-sound \{ (T1, T2). True \}

(proof)

**Lemma** all-relation-is-weak-reduction-bisimulation:

- fixes Cal :: 'a process_calculus
- shows weak-reduction-bisimulation \{ (a, b). True \} Cal

(proof)

**Lemma (in encoding) operational-correspondencemodulo-some-target-relation:**

- shows \( \exists TRel. \) weakly-operational-corresponding \( TRel \)
- \( \exists TRel. \) operational-corresponding \( TRel \)
- \( \exists TRel. \) weakly-operational-corresponding \( TRel \land weak-reduction-bisimulation \( TRel \) Target \)
- \( \exists TRel. \) operational-corresponding \( TRel \land weak-reduction-bisimulation \( TRel \) Target \)

(proof)

Strong operational correspondence requires that source can perform a step iff their translations can perform a step.

**Lemma (in encoding) strong-operational-correspondencemodulo-some-target-relation:**

- shows \( \exists TRel. \) strongly-operational-corresponding \( TRel \)
- \( \exists TRel. \) strongly-operational-corresponding \( TRel \land weak-reduction-bisimulation \( TRel \) Target \)
- \( \exists TRel. \) weakly-operational-sound \( TRel \land weak-reduction-bisimulation \( TRel \) Target \)

(proof)

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 :: \((\text{proc} S, \text{proc} T)\) Proc \& T set
- and TRel :: \((\text{proc} T \times \text{proc} T)\) set
- assumes A1: \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \)
- and A2: \( \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)
- and A3: \( \text{weak-reduction-simulation } \text{Rel} (\text{STCal } \text{Source } \text{Target}) \)

shows operational-complete \( (\text{TRel}^+) \)

(proof)

**Lemma (in encoding) OCom-iff-indRelRTPO-is-weak-reduction-simulation:**

- fixes TRel :: \((\text{proc} T \times \text{proc} T)\) set
- shows operational-complete \( (\text{TRel}^+) \)
  \( \land \) weak-reduction-simulation \( (\text{TRel}^+) \) Target
  \( = \) weak-reduction-simulation \( (\text{indRelRTPO } \text{TRel}) \) (STCal Source Target)

(proof)

**Lemma (in encoding) OCom-iff-weak-reduction-simulation:**

- fixes TRel :: \((\text{proc} T \times \text{proc} T)\) set
- shows operational-complete \( (\text{TRel}^+) \)
  \( \land \) weak-reduction-simulation \( (\text{TRel}^+) \) Target
  \( = \) \( (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \land (\forall \text{T1 T2}, (\text{T1}, \text{T2}) \in \text{TRel} \rightarrow (\text{TargetTerm } \text{T1}, \text{TargetTerm } \text{T2}) \in \text{Rel} \land (\forall \text{T1 T2}, (\text{T1}, \text{T2}) \in \text{TRel}^+) \))
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel —> ([S], T) ∈ TRel*)
∧ weak-reduction-simulation Rel (STCal Source Target))

(proof)

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)

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)

lemma (in encoding) SOCom-iff-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  shows (strongly-operational-complete (TRel*)
    ∧ strong-reduction-simulation (TRel *) Target)
    = STCal Source Target)

(proof)

lemma (in encoding) target-relation-from-source-target-relation:
  assumes str: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel
      —> (TargetTerm ([S]), TargetTerm T) ∈ Rel*
  shows ∃ TRel. (∀ 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*)

(proof)

lemma (in encoding) SOCom-modulo-TRel-iff-strong-reduction-simulation:
  shows (∃ TRel. strongly-operational-complete (TRel*)
    ∧ strong-reduction-simulation (TRel*) Target)
    = STCal Source Target)

(proof)

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 $A_2$: $\forall S,T.\ (\mathrm{SourceTerm}\ S\ ,\ \mathrm{TargetTerm}\ T) \in \text{Rel} \rightarrow ([S],T) \in \text{TRel}^*$
and $A_3$: weak-reduction-contrasimulation ($\text{Rel}^{-1}$) ($\text{STCal}\ Source\ Target$)
shows weakly-operational-sound ($\text{TRel}^*$)
(proof)

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 × (procS,'procT)' Proc) set
and TRel :: (procT × 'procT)' set
assumes $A_1$: $\forall S\ (\mathrm{SourceTerm}\ S\ ,\ \mathrm{TargetTerm}\ ([S])\ ) \in \text{Rel}$
and $A_2$: $\forall S\ T\ (\mathrm{SourceTerm}\ S\ ,\ \mathrm{TargetTerm}\ T) \in \text{Rel} \rightarrow ([S],T) \in \text{TRel}^*$
and $A_3$: weak-reduction-simulation ($\text{Rel}^{-1}$) ($\text{STCal}\ Source\ Target$)
shows operational-sound ($\text{TRel}^*$)
(proof)

**Lemma (in encoding) OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation:**
fixes TRel :: (procT × 'procT)' set
shows (operational-sound ($\text{TRel}^*$))
∧ weak-reduction-simulation (($\text{TRel}^*)^{-1}$) Target
= weak-reduction-simulation (($\text{indRelRTPO\ TRel})^{-1}$) ($\text{STCal}\ Source\ Target$)
(proof)

**Lemma (in encoding) OSou-iff-weak-reduction-simulation:**
fixes TRel :: (procT × 'procT)' set
shows (operational-sound ($\text{TRel}^*$))
∧ weak-reduction-simulation (($\text{TRel}^*)^{-1}$) Target
= weak-reduction-simulation ($\text{Rel}^{-1}$) ($\text{STCal}\ Source\ Target$)
(proof)

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 × (procS,'procT)' Proc) set
and TRel :: (procT × 'procT)' set
assumes $A_1$: $\forall S\ (\mathrm{SourceTerm}\ S\ ,\ \mathrm{TargetTerm}\ ([S])\ ) \in \text{Rel}$
and $A_2$: $\forall S\ T\ (\mathrm{SourceTerm}\ S\ ,\ \mathrm{TargetTerm}\ T) \in \text{Rel} \rightarrow ([S],T) \in \text{TRel}^*$
and $A_3$: strong-reduction-simulation ($\text{Rel}^{-1}$) ($\text{STCal}\ Source\ Target$)
shows strongly-operational-sound ($\text{TRel}^*$)
(proof)

**Lemma (in encoding) SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation:**
fixes TRel :: (procT × 'procT)' set
shows (strongly-operational-sound ($\text{TRel}^*$))
∧ strong-reduction-simulation (($\text{TRel}^*)^{-1}$) Target
= strong-reduction-simulation (($\text{indRelRTPO\ TRel})^{-1}$) ($\text{STCal}\ Source\ Target$)
(proof)

**Lemma (in encoding) SOSou-iff-strong-reduction-simulation:**
fixes TRel :: (procT × 'procT)' set
shows (strongly-operational-sound ($\text{TRel}^*$) ∧ strong-reduction-simulation (($\text{TRel}^*)^{-1}$) Target)
\[
\begin{align*}
\text{lemma (in encoding)} & \text{ S\textsubscript{S\textsubscript{O\textsubscript{S\textsubscript{O}} - modulo-TRel}} - iff - strong - reduction - simulation:} \\
\text{shows} & \exists \text{ TRel. strongly-operational-sound (TRel\textsuperscript{*})} \\
\text{and} & \text{ TRel -iff-strong-reduction-simulation} \\
= & \exists \text{ Rel. (S, T) \in Rel} \\
\wedge & \forall T1 T2. (T1, T2) \in TRel \rightarrow (T1, T2) \in TRel\textsuperscript{*} \\
\wedge & \forall S T. (S, T) \in TRel \rightarrow (S, T) \in TRel\textsuperscript{*} \\
\wedge & \text{ weak - reduction - correspondence - simulation (TRel\textsuperscript{*}) (STCal Source Target)} \\
\langle \text{proof} \rangle
\end{align*}
\]

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.

\textbf{lemma (in encoding) WOC-iff-indRelRTPO-is-reduction-correspondence-simulation:}

\textbf{fixes} \text{ Rel :: } (\text{procS, } \text{procT}) \times (\text{procS, } \text{procT}) \text{ set}
\text{ and } \text{ TRel :: } (\text{procT} \times \text{procT}) \text{ set}
\text{ assumes enc: } \forall S. (S, S) \in Rel \\
\text{ and } \text{ TRel :: } (S, S) \in TRel\textsuperscript{*} \\
\text{ and } \text{ cs: weak-reduction-correspondence-simulation Rel (STCal Source Target)}
\text{ shows weakly-operational-corresponding (TRel\textsuperscript{*})}
\langle \text{proof} \rangle

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.

\textbf{lemma (in encoding) WOC-iff-indRelRTPO-is-reduction-correspondence-simulation:}

\textbf{fixes} \text{ TRel :: } (\text{procT} \times \text{procT}) \text{ set}
\text{ shows (weakly-operational-corresponding (TRel\textsuperscript{*})} \\
\text{ weak - reduction - correspondence - simulation (TRel\textsuperscript{*}) Target)} \\
= \text{ weak - reduction - correspondence - simulation (indRelRTPO TRel) (STCal Source Target)} \\
\langle \text{proof} \rangle

\textbf{lemma (in encoding) WOC-iff-reduction-correspondence-simulation:}

\textbf{fixes} \text{ TRel :: } (\text{procT} \times \text{procT}) \text{ set}
\text{ shows (weakly-operational-corresponding (TRel\textsuperscript{*})} \\
\text{ weak - reduction - correspondence - simulation (TRel\textsuperscript{*}) Target)} \\
= \text{ weak - reduction - correspondence - simulation (indRelRTPO TRel) (STCal Source Target)} \\
\langle \text{proof} \rangle

\textbf{lemma rel-includes-TRel-modulo-preorder:}

\textbf{fixes} \text{ Rel :: } (\text{procS, } \text{procT}) \times (\text{procS, } \text{procT}) \text{ set}
\text{ and } \text{ TRel :: } (\text{procT} \times \text{procT}) \text{ set}
\text{ assumes transT: trans TRel}
\text{ shows } ((T1, T2) \in TRel \rightarrow (T1, T2) \in TRel\textsuperscript{*}) \\
\text{ and } \text{ TRel :: } (T1, T2) \in TRel\textsuperscript{*} \\
= ((T1, T2) \in TRel = \{ (T1, T2) \}) \\
\langle \text{proof} \rangle
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 indRelRTO, 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-wrt-indRelRTO-is-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  shows (operational-corresponding (TRel⁺) ∧ weak-reduction-bisimulation (TRel⁺) Target)
    = weak-reduction-bisimulation (indRelRTO TRel) (STCal Source Target)
  (proof)

lemma (in encoding) OC-wrt-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  shows (operational-corresponding (TRel⁺) ∧ weak-reduction-bisimulation (TRel⁺) Target)
    = (∃ Rel. (∀ S T. (SourceTerm S, TargetTerm (S)) ∈ Rel)
      ∧ (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
      ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel⁺)
      ∧ weak-reduction-bisimulation Rel (STCal Source Target))
  (proof)

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 T. (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)
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 × 'procT) set**
- **shows (strongly-operational-corresponding (TRel*)**
  \(\land\) strong-reduction-bisimulation (TRel*) Target)
  = strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

  \(\langle proof\rangle\)

**lemma (in encoding) SOC-iff-strong-reduction-bisimulation:**

- **fixes TRel :: ('procT × 'procT) set**
- **shows (strongly-operational-corresponding (TRel*)**
  \(\land\) strong-reduction-bisimulation (TRel*) Target)
  = (\(\exists\) Rel. (\(\forall\) S. (SourceTerm S, TargetTerm ([S])) \(\in\) Rel)
  \(\land\) TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \(\in\) Rel\})
  \(\land\) (\(\forall\) S T. (SourceTerm S, TargetTerm T) \(\in\) Rel \(\rightarrow\) ([S], T) \(\in\) TRel* )
  \(\land\) strong-reduction-bisimulation Rel (STCal Source Target))

  \(\langle proof\rangle\)

**lemma (in encoding) SOC-wrt-preorder-iff-strong-reduction-bisimulation:**

- **fixes TRel :: ('procT × 'procT) set**
- **shows (strongly-operational-corresponding TRel \(\land\) preorder TRel**
  \(\land\) strong-reduction-bisimulation TRel Target)
  = (\(\exists\) Rel. (\(\forall\) S. (SourceTerm S, TargetTerm ([S])) \(\in\) Rel)
  \(\land\) TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \(\in\) Rel\})
  \(\land\) (\(\forall\) S T. (SourceTerm S, TargetTerm T) \(\in\) Rel \(\rightarrow\) ([S], T) \(\in\) TRel )
  \(\land\) preorder Rel
  \(\land\) strong-reduction-bisimulation Rel (STCal Source Target))

  \(\langle proof\rangle\)

**lemma (in encoding) SOC-wrt-TRel-iff-strong-reduction-bisimulation:**

- **shows (\(\exists\) TRel. strongly-operational-corresponding (TRel*)**
  \(\land\) strong-reduction-bisimulation (TRel*) Target)
  = (\(\exists\) Rel. (\(\forall\) S. (SourceTerm S, TargetTerm ([S])) \(\in\) Rel)
  \(\land\) TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \(\in\) Rel\})
  \(\land\) (\(\forall\) S T. (SourceTerm S, TargetTerm T) \(\in\) Rel \(\rightarrow\) ([S], TargetTerm T) \(\in\) Rel* )
  \(\land\) strong-reduction-bisimulation Rel (STCal Source Target))

  \(\langle proof\rangle\)

**lemma (in encoding) SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation:**

- **shows (strongly-operational-corresponding TRel \(\land\) strong-reduction-bisimulation TRel Target)
  \(\iff\) strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)

  \(\langle proof\rangle\)

**lemma (in encoding) SOC-wrt-equivalence-iff-strong-reduction-bisimulation:**

- **shows (strongly-operational-corresponding TRel \(\land\) strong-reduction-bisimulation TRel Target)
  \(\iff\) (\(\exists\) Rel. (\(\forall\) S. (SourceTerm S, TargetTerm ([S])) \(\in\) Rel
  \(\land\) TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \(\in\) Rel\})
  \(\land\) trans Rel \(\land\) strong-reduction-bisimulation Rel (STCal Source Target))

  \(\langle proof\rangle\)

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 × ′procS) set ⇒ (′procT × ′procT) set ⇒ bool
  where
    fully-abstract SRel TRel ≡ ∀ S1 S2. (S1, S2) ∈ SRel ←→ ([S1], [S2]) ∈ 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: ∀ S1 S2. [S1] = [S2] −→ S1 = S2
  shows fully-abstract {(S1, S2). S1 = S2} {(T1, T2). T1 = T2}
  ⟨proof⟩

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 {} {}
  ⟨proof⟩

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}
  ⟨proof⟩

If the encoding is injective then for each source term relation RelS there exists a target term relation
RelT such that the encoding is fully abstract w.r.t. RelS and RelT.

lemma (in encoding) fully-abstract-wrt-source-relation:
  fixes RelS :: (′procS × ′procS) set
  assumes injectivity: ∀ S1 S2. [S1] = [S2] −→ S1 = S2
  shows ∃ RelT. fully-abstract RelS RelT
  ⟨proof⟩

If all source terms that are translated to the same target term are related by a trans source term
relation RelS, then there exists a target term relation RelT such that the encoding is fully abstract
w.r.t. RelS and RelT.

lemma (in encoding) fully-abstract-wrt-trans-source-relation:
  fixes RelS :: (′procS × ′procS) set
  assumes encRelS: ∀ S1 S2. [S1] = [S2] −→ (S1, S2) ∈ RelS
      and transS: trans RelS
  shows ∃ RelT. fully-abstract RelS RelT
  ⟨proof⟩

lemma (in encoding) fully-abstract-wrt-trans-closure-of-source-relation:
  fixes RelS :: (′procS × ′procS) set
  assumes encRelS: ∀ S1 S2. [S1] = [S2] −→ (S1, S2) ∈ RelS+
  shows ∃ RelT. fully-abstract (RelS+) RelT
  ⟨proof⟩
For every encoding and every target term relation \( \text{RelT} \) there exists a source term relation \( \text{RelS} \) such that the encoding is fully abstract w.r.t. \( \text{RelS} \) and \( \text{RelT} \).

**Lemma (in encoding) fully-abstract-wrt-target-relation:**

- **fixes** \( \text{RelT} :: (\text{Proc} \times \text{Proc}) \) set
- **shows** \( \exists \text{RelS}, \) fully-abstract \( \text{RelS} \) \( \text{RelT} \)

(\text{proof})

### 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 \( \text{SRel} \) and \( \text{TRel} \) we can construct a relation on the disjunctive union of source and target terms, whose reduction to source terms is \( \text{SRel} \) and whose reduction to target terms is \( \text{TRel} \). But full abstraction ensures that each trans relation that relates source terms and their literal translations in both directions includes \( \text{SRel} \) iff it includes \( \text{TRel} \) restricted to translated source terms.

**Lemma (in encoding) full-abstraction-and-trans-relation-impl-SRel:**

- **fixes** \( \text{Rel} :: ((\text{Proc}, \text{Proc}) \text{ Proc} \times (\text{Proc}, \text{Proc}) \text{ Proc} \text{ set})
- **and** \( \text{SRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **and** \( \text{TRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **assumes** \( \text{fullAbs}: \) fully-abstract \( \text{SRel} \) \( \text{TRel} \)

(\text{proof})

**Lemma (in encoding) full-abstraction-and-trans-relation-impl-TRel:**

- **fixes** \( \text{Rel} :: ((\text{Proc}, \text{Proc}) \text{ Proc} \times (\text{Proc}, \text{Proc}) \text{ Proc} \text{ set})
- **and** \( \text{SRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **and** \( \text{TRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **assumes** \( \text{fullAbs}: \) fully-abstract \( \text{SRel} \) \( \text{TRel} \)

(\text{proof})

**Lemma (in encoding) full-abstraction-and-trans-relation-impl-SRel-iff-TRel:**

- **fixes** \( \text{Rel} :: ((\text{Proc}, \text{Proc}) \text{ Proc} \times (\text{Proc}, \text{Proc}) \text{ Proc} \text{ set})
- **and** \( \text{SRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **and** \( \text{TRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **assumes** \( \text{fullAbs}: \) fully-abstract \( \text{SRel} \) \( \text{TRel} \)

(\text{proof})

**Lemma (in encoding) full-abstraction-and-trans-relation-impl-SRel-encRL:**

- **fixes** \( \text{Rel} :: ((\text{Proc}, \text{Proc}) \text{ Proc} \times (\text{Proc}, \text{Proc}) \text{ Proc} \text{ set})
- **and** \( \text{SRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **and** \( \text{TRel} :: (\text{Proc} \times \text{Proc}) \text{ set})
- **assumes** \( \text{fullAbs}: \) fully-abstract \( \text{SRel} \) \( \text{TRel} \)

(\text{proof})
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)

**Proof**

**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**

**Lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-sym:**

- **Fixes:** SRel :: (procS × procS) set
- **And:** TRel :: (procT × procT) set
- **Assumes:** fullAbs: fully-abstract SRel TRel
- **Shows:** sym SRel ↔ sym {[T1, T2]. ∃ S1 S2. T1 = [S1] ∧ T2 = [S2] ∧ (T1, T2) ∈ TRel}

**Proof**

**Lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-sym:**

- **Fixes:** SRel :: (procS × procS) set
- **And:** TRel :: (procT × procT) set
- **Assumes:** fullAbs: fully-abstract SRel TRel
- **And:** surj: ∀ T. ∃ S. T = [S]
- **Shows:** sym SRel ↔ sym TRel

**Proof**

**Lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-trans:**

- **Fixes:** SRel :: (procS × procS) set
- **And:** TRel :: (procT × procT) set
- **Assumes:** fullAbs: fully-abstract SRel TRel
- **Shows:** trans SRel ↔ trans {[T1, T2]. ∃ S1 S2. T1 = [S1] ∧ T2 = [S2] ∧ (T1, T2) ∈ TRel}

**Proof**

**Lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-trans:**

- **Fixes:** SRel :: (procS × procS) set
- **And:** TRel :: (procT × procT) set
- **Assumes:** fullAbs: fully-abstract SRel TRel
- **And:** surj: ∀ T. ∃ S. T = [S]
- **Shows:** trans SRel ↔ trans TRel

**Proof**

Similarly, a fully abstract encoding that respects a predicate ensures the this predicate is preserved, reflected, or respected by SRel if 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 × 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-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**
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). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
  ⇔ rel-preserves-binary-pred {(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩

lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects:
  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-reflects-binary-pred {(P, Q). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
  ⇔ rel-reflects-binary-pred {(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflection:
  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-reflection-binary-pred {(P, Q). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
  ⇔ rel-reflection-binary-pred {(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflection:
  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-reflection-binary-pred {(P, Q). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
  ⇔ rel-reflection-binary-pred {(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩

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)

lemma (in encoding) source-target-relation-impl-full-abstraction-wrt-trans-closures:
fixes Rel :: (('procS', 'procT') Proc × ('procS', 'procT') Proc) set
and SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes enc: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel
and srel: SRel = {(S1, S2), (SourceTerm S1, SourceTerm S2) ∈ Rel}
and trel: TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}
and trans: ∀ P Q R. (P, Q) ∈ Rel ∧ (Q, R) ∈ Rel ∧ (P ∈ ProcS ∧ Q ∈ ProcT)
∧ (P ∈ ProcT ∧ Q ∈ ProcS) → (P, R) ∈ Rel
shows fully-abstract SRel TRel

(proof)

If an encoding is fully abstract w.r.t. SRel and TRel, then we can conclude from a pair in indRelRTPO or indRelSTEQ on a pair in TRel and SRel.

lemma (in encoding) full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and P Q :: ('procS, 'procT) Proc
assumes fullAbs: fully-abstract SRel TRel
and rel: P ≤[1]R< SRel, TRel> Q
shows ∀ SP SQ. SP ∈ S P ∧ SQ ∈ S Q → ([SP], [SQ]) ∈ TRel∗
and ∀ SP TQ. SP ∈ S P ∧ TQ ∈ T Q → ([SP], TQ) ∈ TRel∗

(proof)

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 trans: trans TRel
and refS: refl SRel
and rel: P ~[1]< 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

(proof)

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 ⟩

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 ⟩

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: ∀ 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 ⟩

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 ⟩

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)

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)

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)

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)

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)

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 \( SRel :: \left( \text{'procS} \times \text{'procS} \right) \set \)
and \( TRel :: \left( \text{'procT} \times \text{'procT} \right) \set \)
and \( Rel :: \left( \text{'procS}, \text{'procT} \right) \text{Proc} \times \left( \text{'procS}, \text{'procT} \right) \text{Proc} \set \)
assumes fully-abstract \( SRel TRel \)
and \( \text{rel} :: \text{Rel} = ((\text{indRelSTEQ} \ SRel \ TRel)) \)
\[- (\{(P, Q). (P \in \text{ProcS} \land Q \in \text{ProcS}) \lor (P \in \text{ProcT} \land Q \in \text{ProcT})) \}
\cup (\{(P, Q). (\exists SP SQ. SP \in S \land SQ \in S \land (SP, SQ) \in SRel) \lor (3 TP TQ. TP \in T \land TQ \in T \land (TP, TQ) \in TRel)) \}
shows (refl \ SRel \land \text{trans} \ TRel) = \text{trans} \ Rel \)
(proof)

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 \( Rel :: \left( \text{'procS}, \text{'procT} \right) \text{Proc} \times \left( \text{'procS}, \text{'procT} \right) \text{Proc} \set \)
and \( SRel :: \left( \text{'procS} \times \text{'procS} \right) \set \)
and \( TRel :: \left( \text{'procT} \times \text{'procT} \right) \set \)
assumes \text{enc} :: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \land (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel}
and \( \text{srel} :: SRel = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\} \)
and \( \text{trel} :: TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\} \)
and \( \text{trans} :: \text{trans} \text{Rel} \)
says fully-abstract \( SRel TRel \)
(proof)

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 \( Rel :: \left( \text{'procS}, \text{'procT} \right) \text{Proc} \times \left( \text{'procS}, \text{'procT} \right) \text{Proc} \set \)
assumes \text{enc} :: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \land (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel}
and \( \text{trans} :: \text{trans} (\text{Rel} \cup \{(P, Q). \exists S. [S] \in T \land P \lor S \in S \land Q)) \}
says fully-abstract \( SRel TRel \)
\{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
(proof)

lemma (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-is-transB:
fixes \( SRel :: \left( \text{'procS} \times \text{'procS} \right) \set \)
and \( TRel :: \left( \text{'procT} \times \text{'procT} \right) \set \)
assumes \text{preord} :: \text{preorder} \ TRel
saying fully-abstract \( SRel TRel \)
(proof)

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 \( SRel :: \left( \text{'procS} \times \text{'procS} \right) \set \)
and \( TRel :: \left( \text{'procT} \times \text{'procT} \right) \set \)
says (fully-abstract \( SRel TRel \) and refl \( SRel \) and \( \text{trans} \ TRel \)) =
\( (\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \land SRel = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}) \land TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}) \land \text{trans} (\text{Rel} \cup \{(P, Q). \exists S. [S] \in T \land P \lor S \in S \land Q)) \)
(proof)
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, 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`
- `assumes Cond enc = (3 Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ Pred Rel)`
- `shows Cond enc = (3 Rel'. Pred (indRelR ∪ Rel'))`

**(proof)**

**Lemma** (in encoding) combine-conditions-on-pairs-of-relations:
- `fixes RelA RelB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set`
- `assumes ∀ (P, Q) ∈ RelA. CondA (P, Q)`
- `and ∀ (P, Q) ∈ RelB. CondB (P, Q)`
- `shows ∀ (P, Q) ∈ RelA ∩ RelB. CondA (P, Q) ∧ (∀ (P, Q) ∈ RelA ∩ RelB. CondB (P, Q))`

**(proof)**

**Lemma** (in encoding) combine-conditions-on-sets-of-relations:
- `assumes ∀ (P, Q) ∈ RelA. CondA (P, Q)`
- `and Cond Rel ∧ Rel ⊆ RelA`
- `shows Cond Rel ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q))`

**(proof)**

**Lemma** (in encoding) combine-conditions-on-sets-and-pairs-of-relations:
- `fixes RelA RelB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set`
- `assumes ∀ (P, Q) ∈ RelA. CondA (P, Q)`
- `and ∀ (P, Q) ∈ RelB. CondB (P, Q)`
- `and Cond Rel ∧ Rel ⊆ RelA ∧ Rel ⊆ RelB`
- `shows Cond Rel ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q)) ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q))`

**(proof)**

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 × ('procS, 'procT) Proc) set`
- `assumes A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ RelA`
- `and A2: ∀ (P, Q) ∈ RelA. CondA (P, Q) 77`
and \( A3 \): \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{RelB} \)
and \( A4 \): \( \forall (P, Q) \in \text{RelB}. \text{CondB} (P, Q) \)
shows \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{CondA} (P, Q)) \land (\forall (P, Q) \in \text{Rel}. \text{CondB} (P, Q)) \)
and \( \text{Cond indRelR} \implies (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{CondA} (P, Q)) \land (\forall (P, Q) \in \text{Rel}. \text{CondB} (P, Q)) \land \text{Cond Rel} \)

\( \langle \text{proof} \rangle \)

**Lemma (in encoding) indRelR-cond-respects-predA-and-reflects-predB:**

fixes \( \text{PredA} \) \( \text{PredB} \) :: \( \langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \text{bool} \)
shows \( (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-respects-pred Rel PredA} \land (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-reflects-pred Rel PredB}) \)
\( = (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-respects-pred Rel PredA} \land \text{rel-reflects-pred Rel PredB}) \)

\( \langle \text{proof} \rangle \)

### 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 \( \text{success} :: \langle \text{procS}, \text{procT} \rangle \text{Proc} \)
shows \( \langle \text{enc-weakly-respects-barb-set} \{ \text{success} \} \land \text{enc-reflects-divergence} \rangle \)
\( = (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{ \text{success} \} \land \text{rel-reflects-divergence Rel} (\text{STCal Source Target})) \)

\( \langle \text{proof} \rangle \)

**Lemma (in encoding-wrt-barbs) SS-DR-iff-source-target-rel:**

fixes \( \text{success} :: \langle \text{procS}, \text{procT} \rangle \text{Proc} \)
shows \( \langle \text{enc-respects-barb-set} \{ \text{success} \} \land \text{enc-reflects-divergence} \rangle \)
\( = (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{ \text{success} \} \land \text{rel-reflects-divergence Rel} (\text{STCal Source Target})) \)

\( \langle \text{proof} \rangle \)

### 10.2 Adding Operational Correspondence

The effect of operational correspondence includes conditions (TRel is included, transitivity) that require a witness like \( \text{indRelRTPO} \). In order to combine operational correspondence with success sensitiveness, we show that if the encoding and TRel (weakly) respects barbs than \( \text{indRelRTPO} \) (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 \( \text{success} :: \langle \text{procS}, \text{procT} \rangle \text{Proc} \)
and \( \text{TRel} :: \langle \text{procT} \times \text{procT} \rangle \text{set} \)
assumes \( \text{encRS: enc-weakly-respects-barb-set} \{ \text{success} \} \)
and \( \text{trelPS: rel-weakly-preserves-barb-set} \text{TRel TWB} \{ \text{success} \} \)
and \( \text{trelRS: rel-weakly-reflects-barb-set} \text{TRel TWB} \{ \text{success} \} \)
shows \( \text{rel-weakly-respects-barbs} \langle \text{indRelRTPO TRel} \rangle (\text{STCalWB SWB TWB}) \{ \text{success} \} \)

\( \langle \text{proof} \rangle \)

**Lemma (in encoding-wrt-barbs) enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs:**

fixes \( \text{TRel} :: \langle \text{procT} \times \text{procT} \rangle \text{set} \)
assumes \( \text{encRS: enc-weakly-respects-barbs} \)
and \( \text{trelPS: rel-weakly-preserves-barbs} \text{TRel TWB} \)
and \( \text{trelRS: rel-weakly-reflects-barbs} \text{TRel TWB} \)
shows \( \text{rel-weakly-respects-barbs} \langle \text{indRelRTPO TRel} \rangle (\text{STCalWB SWB TWB}) \)
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)

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)

An encoding is success sensitive and operational corresponding w.r.t. a bisimulation TRel that respects success iff there exists a bisimulation that includes TRel and respects success. The same holds if we consider not only success sensitiveness but barb sensitiveness in general.

lemma (in encoding-wrt-barbs) OC-SS-iff-source-target-rel:
  fixes success :: 'barbs
  and TRel :: ('procT × 'procT) set
  shows (operational-corresponding (TRel))
    ∧ weak-reduction-bisimulation (TRel*) Target
    ∧ enc-weakly-respects-barb-set {success}
    ∧ rel-weakly-respects-barb-set TRel TWB {success})
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([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)

lemma (in encoding-wrt-barbs) OC-SS-RB-iff-source-target-rel:
  fixes success :: 'barbs
  and TRel :: ('procT × 'procT) set
  shows (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})
  = (∃ 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})
(proof)

lemma (in encoding-wrt-barbs) OC-SS-wpreorder-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)
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 T. (SourceTerm S, TargetTerm T) ∈ Rel → (STS, T) ∈ TRel)
∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success})
(proof)

An encoding is success sensitive and weakly operational corresponding w.r.t. a correspondence simulation TRel that respects success iff there exists a correspondence simulation that includes TRel and respects success. The same holds if we consider not only success sensitiveness but barb sensitiveness in general.

lemma (in encoding-wrt-barbs) WOC-SS-wrt-preorder-iff-source-target-rel:

fixes success :: 'barbs
and TRel :: ('procT × 'procT) set
shows (weakly-operational-corresponding TRel ∧ preorder TRel
∧ weak-reduction-correspondence-simulation TRel Target
∧ enc-weakly-respects-barb-set {success}
∧ rel-weakly-respects-barb-set TRel TWB {success})
= (∃ Rel. (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → (STS, T) ∈ TRel)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target) ∧ preorder Rel
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success})
(proof)

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 TRel TWB
∧ rel-weakly-respects-barbs TWB {success})
= (∃ Rel. (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → (STS, 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)

An encoding is strongly success sensitive and strongly operational corresponding w.r.t. a strong bisimulation TRel that strongly respects success iff there exists a strong bisimulation that includes TRel and strongly respects success. The same holds if we consider not only strong success sensitiveness but strong barb sensitiveness in general.

lemma (in encoding-wrt-barbs) SOC-SS-wrt-preorder-iff-source-target-rel:

fixes success :: 'barbs
Next we also add divergence reflection to operational correspondence and success sensitiveness.

**Lemma** (in encoding-wrt-barbs) **SOC-SS-RB-wrt-preorder-iff-source-target-rel:**

```plaintext
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})
= (∃ SRel. (∀ 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)
∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success})
```

(proof)

**Lemma** (in encoding) **enc-and-TRelimpl-indRelRTPO-reflect-divergence:**

```plaintext
fixes TRel :: ('procT × 'procT) set

assumes encRD: enc-reflects-divergence

and trelRD: rel-reflects-divergence TRel Target

shows rel-reflects-divergence (indRelRTPO TRel) (STCal Source Target)
```

(proof)

**Lemma** (in encoding-wrt-barbs) **OC-SS-DR-iff-source-target-rel:**

```plaintext
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}
∧ 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)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ 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)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
∧ rel-weakly-respects-barb-set Rel (STCal Source Target))
```

(proof)

**Lemma** (in encoding-wrt-barbs) **WOC-SS-DR-wrt-preorder-iff-source-target-rel:**

```plaintext
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)
= (∃ S. (SourceTerm S, TargetTerm (([S]) ∈ Rel)
∧ TRel = {((T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
```
\[\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \implies ([S], T) \in \text{TRel}\]
\[\text{weak-reduction-correspondence-simulation } \text{Rel (STCal Source Target) \land preorder } \text{Rel}\]
\[\text{rel-weakly-respects-barb-set } \text{Rel (STCalWB SWB TWB) \{success\}}\]
\[\text{rel-reflected-divergence } \text{Rel (STCal Source Target)\}}\]

\text{\textsf{(proof)}}

\text{\textbf{lemma (in encoding-wrt-barbs) OC-SS-DR-wrt-preorder-iff-source-target-rel:}}
\begin{itemize}
\item \text{\texttt{fixes success :: \textquote{\textsc{bars}}}}
\item \text{\texttt{and TRel :: \textquote{\textsc{procT} \times \textsc{procT} set}}}
\item \text{\texttt{shows (operational-corresponding } TRel \land preorder } TRel \land \text{weak-reduction-bisimulation } TRel \text{Source Target}}
\item \text{\texttt{\land enc-weakly-respects-barb-set \{success\}}}\]
\item \text{\texttt{\land enc-reflected-divergence \land rel-reflected-divergence } TRel \text{Target}}
\end{itemize}
\[= (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\]
\[\land TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}\]
\[\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \implies ([S], T) \in \text{TRel} )\]
\[\land \text{weak-reduction-bisimulation Rel (STCal Source Target) \land preorder Rel}\]
\[\land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}\]
\[\land \text{rel-reflected-divergence Rel (STCal Source Target)\}}\]

\text{\textsf{(proof)}}

\text{\textbf{lemma (in encoding-wrt-barbs) SOC-SS-DR-wrt-preorder-iff-source-target-rel:}}
\begin{itemize}
\item \text{\texttt{fixes success :: \textquote{\textsc{bars}}}}
\item \text{\texttt{and TRel :: \textquote{\textsc{procT} \times \textsc{procT} set}}}
\item \text{\texttt{shows (strongly-operational-corresponding } TRel \land preorder } TRel \land \text{weak-reduction-bisimulation } TRel \text{Source Target}}
\item \text{\texttt{\land enc-respects-barb-set \{success\} \land rel-respects-barb-set } TRel \text{TWB \{success\}}}\]
\item \text{\texttt{\land enc-reflected-divergence \land rel-reflected-divergence } TRel \text{Source Target}}
\end{itemize}
\[= (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\]
\[\land TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}\]
\[\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \implies ([S], T) \in \text{TRel} )\]
\[\land \text{weak-reduction-bisimulation Rel (STCal Source Target) \land preorder Rel}\]
\[\land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}\]
\[\land \text{rel-reflected-divergence Rel (STCal Source Target)\}}\]

\text{\textsf{(proof)}}

\textbf{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.

\textbf{lemma (in encoding) FS-OC-modulo-equivalences-iff-source-target-relation:}
\begin{itemize}
\item \text{\texttt{fixes SRel :: \textquote{\textsc{procS} \times \textsc{procS} set}}}
\item \text{\texttt{and TRel :: \textquote{\textsc{procT} \times \textsc{procT} set}}}
\item \text{\texttt{assumes eqS: equivalence SRel}}
\item \text{\texttt{and eqT: equivalence TRel}}
\item \text{\texttt{shows fully-abstract SRel TRel}}
\item \text{\texttt{\land operational-corresponding } TRel \land \text{weak-reduction-bisimulation } TRel \text{Source Target}}
\item \text{\texttt{\iff (\exists \text{Rel.}}\]
\item \text{\texttt{\land (\forall S. (SourceTerm S, TargetTerm ([S])) \in \text{Rel} \land (TargetTerm ([S]), SourceTerm S) \in \text{Rel})}}\]
\item \text{\texttt{\land SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in \text{Rel}\}}}\]
\item \text{\texttt{\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in \text{Rel}\}}}\]
\item \text{\texttt{\land \trans \text{Rel} \land \text{weak-reduction-bisimulation Rel (STCal Source Target)}\}}\]
\end{itemize}

\text{\textsf{(proof)}}

\textbf{lemma (in encoding) FA-SOC-modulo-equivalences-iff-source-target-relation:}
\begin{itemize}
\item \text{\texttt{fixes SRel :: \textquote{\textsc{procS} \times \textsc{procS} set}}}
\item \text{\texttt{and TRel :: \textquote{\textsc{procT} \times \textsc{procT} set}}}
\item \text{\texttt{assumes eqS: equivalence SRel}}
\end{itemize}
and \( \text{eqT: equivalence TRel} \)

shows fully-abstract \( \text{SRel} \) \( \text{TRel} \) ∧ strongly-operational-corresponding \( \text{TRel} \)

\[
\land \text{strong-reduction-bisimulation TRel Target} \iff (\exists \text{Rel}).
\]

\[
(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \land (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{Rel})
\]

\[
\land \text{SRel } = \{(S1, S2), (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}
\]

\[
\land \text{TRel } = \{(T1, T2), (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\} \land \text{trans Rel}
\]

\[
\land \text{strong-reduction-bisimulation Rel (STCal Source Target)}
\]

\(\langle \text{proof} \rangle\)

An encoding that is fully abstract w.r.t. the equivalences \( \text{SRel} \) and \( \text{TRel} \) and operationally corresponding w.r.t. \( \text{TRel} \) ensures that \( \text{SRel} \) is a bisimulation iff \( \text{TRel} \) is a bisimulation.

\[\text{lemma (in encoding) FA-and-OC-and-TRel-impl-SRel-bisimulation:}\]

\[\text{fixes SRel :: ('procS x 'procS) set}\]

\[\text{and TRel :: ('procT x 'procT) set}\]

\[\text{assumes fullAbs: fully-abstract SRel TRel}\]

\[\text{and opCom: operational-complete TRel}\]

\[\text{and opSou: operational-sound TRel}\]

\[\text{and symmT: sym TRel}\]

\[\text{and transT: trans TRel}\]

\[\text{and bisimT: weak-reduction-bisimulation TRel Target}\]

shows weak-reduction-bisimulation \( \text{SRel} \) Source

\(\langle \text{proof} \rangle\)

\[\text{lemma (in encoding) FA-and-SOC-and-TRel-impl-SRel-strong-bisimulation:}\]

\[\text{fixes SRel :: ('procS x 'procS) set}\]

\[\text{and TRel :: ('procT x 'procT) set}\]

\[\text{assumes fullAbs: fully-abstract SRel TRel}\]

\[\text{and opCom: strongly-operational-complete TRel}\]

\[\text{and opSou: strongly-operational-sound TRel}\]

\[\text{and symmT: sym TRel}\]

\[\text{and transT: trans TRel}\]

\[\text{and bisimT: strong-reduction-bisimulation TRel Target}\]

shows strong-reduction-bisimulation \( \text{SRel} \) Source

\(\langle \text{proof} \rangle\)

\[\text{lemma (in encoding) FA-and-OC-impl-SRel-iff-TRel-bisimulation:}\]

\[\text{fixes SRel :: ('procS x 'procS) set}\]

\[\text{and TRel :: ('procT x 'procT) set}\]

\[\text{assumes fullAbs: fully-abstract SRel TRel}\]

\[\text{and opCor: operational-corresponding TRel}\]

\[\text{and symmT: sym TRel}\]

\[\text{and transT: trans TRel}\]

\[\text{and surj: } \forall T. \exists S. T = [S]\]

shows weak-reduction-bisimulation \( \text{SRel} \) Source \( \iff \) weak-reduction-bisimulation \( \text{TRel} \) Target

\(\langle \text{proof} \rangle\)

end