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
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 ∪) = (symcl Rel)^
  ⟨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.
The reflexive closure of a symmetric relation is symmetric.

**Lemma** refl-symm-trans-closure-is-symm-refl-trans-closure:

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Shows* $(\text{symcl (Rel})^+ = (\text{symcl Rel})^+$

(*Proof*)

The reflexive closure of a symmetric relation is symmetric.

**Lemma** sym-reflcl-of-symm-rel:

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Assumes* $\text{sym Rel}$

*Shows* $\text{sym (Rel}= Rel)

(*Proof*)

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

**Lemma** reflcl-of-refl-rel:

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Assumes* $\text{refl Rel}$

*Shows* $\text{Rel}= Rel$

(*Proof*)

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

**Lemma** symm-closure-of-symm-rel:

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Assumes* $\text{sym Rel}$

*Shows* $\text{symcl Rel} = Rel$

(*Proof*)

The reflexive and transitive closure of a preorder $\text{Rel}$ is $\text{Rel}$.

**Lemma** rtrancl-of-preorder:

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Assumes* preorder $\text{Rel}$

*Shows* $\text{Rel}= Rel$

(*Proof*)

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* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Shows* $\text{Rel} \subseteq (\text{symcl (Rel})^+ = (\text{symcl Rel})^+$

(*Proof*)

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

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

*Fixes* $\text{Rel} :: (\mathcal{P} a \times \mathcal{P} a)$ set

*Assumes* $\text{condA: } \forall a b c. (a, b) \in \text{Rel} \land (c, b) \in \text{Rel} \land (a, c) \notin \text{Rel}$

$\rightarrow (b, a) \in \text{Rel} \lor (b, c) \in \text{Rel}$

*and* $\text{condB: } \forall a b c. (a, b) \in \text{Rel} \land (a, c) \in \text{Rel} \land (b, c) \notin \text{Rel}$

$\rightarrow (b, a) \in \text{Rel} \lor (c, a) \in \text{Rel}$

*and* reflR: refl $\text{Rel}$

*and* tranR: tran $\text{Rel}$

*Shows* $\text{trans (symcl Rel)}$

(*Proof*)

1.2 Preservation, Reflection, and Respection of Predicates

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

**Abbreviation** rel-preserves-pred :: $(\mathcal{P} a \times \mathcal{P} a)$ set $\Rightarrow (\mathcal{P} a \Rightarrow \text{bool}) \Rightarrow \text{bool}$ where
relation preserves predicate

\[
\text{rel-preserves-pred } \text{Rel } \text{Pred} \equiv \forall a \ b. \ (a, b) \in \text{Rel} \land \text{Pred } a \rightarrow \text{Pred } b
\]

abbreviation rel-preserves-binary-pred :: \((a \times a)\) set \Rightarrow \((a \Rightarrow b \Rightarrow \text{bool})\) \Rightarrow \text{bool} where

\[
\text{rel-preserves-binary-pred } \text{Rel } \text{Pred} \equiv \forall a \ b \ x. \ (a, b) \in \text{Rel} \land \text{Pred } a \ x \rightarrow \text{Pred } b \ x
\]

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

abbreviation rel-reflects-pred :: \((a \times a)\) set \Rightarrow \((a \Rightarrow \text{bool})\) \Rightarrow \text{bool} where

\[
\text{rel-reflects-pred } \text{Rel } \text{Pred} \equiv \forall a \ b. \ (a, b) \in \text{Rel} \land \text{Pred } b \rightarrow \text{Pred } a
\]

abbreviation rel-reflects-binary-pred :: \((a \times a)\) set \Rightarrow \((a \Rightarrow b \Rightarrow \text{bool})\) \Rightarrow \text{bool} where

\[
\text{rel-reflects-binary-pred } \text{Rel } \text{Pred} \equiv \forall a \ b \ x. \ (a, b) \in \text{Rel} \land \text{Pred } b \ x \rightarrow \text{Pred } a \ x
\]

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

abbreviation rel-respects-pred :: \((a \times a)\) set \Rightarrow \((a \Rightarrow \text{bool})\) \Rightarrow \text{bool} where

\[
\text{rel-respects-pred } \text{Rel } \text{Pred} \equiv \text{rel-preserves-pred } \text{Rel } \text{Pred} \land \text{rel-reflects-pred } \text{Rel } \text{Pred}
\]

abbreviation rel-respects-binary-pred :: \((a \times a)\) set \Rightarrow \((a \Rightarrow b \Rightarrow \text{bool})\) \Rightarrow \text{bool} where

\[
\text{rel-respects-binary-pred } \text{Rel } \text{Pred} \equiv \text{rel-preserves-binary-pred } \text{Rel } \text{Pred} \land \text{rel-reflects-binary-pred } \text{Rel } \text{Pred}
\]

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

lemma symm-relation-impl-preservation-equals-reflection:

\[
\text{fixes } \text{Rel} :: \ ((a \times a) \text{ set}) \\
\text{and } \text{Pred} :: \ 'a \Rightarrow \text{bool} \\
\text{assumes symm: } \text{sym } \text{Rel} \\
\text{shows } \text{rel-preserves-pred } \text{Rel } \text{Pred} = \text{rel-reflects-pred } \text{Rel } \text{Pred} \\
\text{and } \text{rel-preserves-pred } \text{Rel } \text{Pred} = \text{rel-respects-pred } \text{Rel } \text{Pred} \\
\text{and } \text{rel-reflects-pred } \text{Rel } \text{Pred} = \text{rel-respects-pred } \text{Rel } \text{Pred}
\]

⟨ proof ⟩

lemma symm-relation-impl-preservation-equals-reflection-of-binary-predicates:

\[
\text{fixes } \text{Rel} :: \ ((a \times a) \text{ set}) \\
\text{and } \text{Pred} :: \ 'a \Rightarrow 'b \Rightarrow \text{bool} \\
\text{assumes symm: } \text{sym } \text{Rel} \\
\text{shows } \text{rel-preserves-binary-pred } \text{Rel } \text{Pred} = \text{rel-reflects-binary-pred } \text{Rel } \text{Pred} \\
\text{and } \text{rel-preserves-binary-pred } \text{Rel } \text{Pred} = \text{rel-respects-binary-pred } \text{Rel } \text{Pred} \\
\text{and } \text{rel-reflects-binary-pred } \text{Rel } \text{Pred} = \text{rel-respects-binary-pred } \text{Rel } \text{Pred}
\]

⟨ proof ⟩

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

lemma preservation-and-closures:

\[
\text{fixes } \text{Rel} :: \ ((a \times a) \text{ set}) \\
\text{and } \text{Pred} :: \ 'a \Rightarrow \text{bool} \\
\text{assumes preservation: } \text{rel-preserves-pred } \text{Rel } \text{Pred} \\
\text{shows } \text{rel-preserves-pred } (\text{Rel}=) \text{ Pred} \\
\text{and } \text{rel-preserves-pred } (\text{Rel}^+) \text{ Pred} \\
\text{and } \text{rel-preserves-pred } (\text{Rel}^*) \text{ Pred}
\]

⟨ proof ⟩

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

\[
\text{fixes } \text{Rel} :: \ ((a \times a) \text{ set}) \\
\text{and } \text{Pred} :: \ 'a \Rightarrow 'b \Rightarrow \text{bool} \\
\text{assumes preservation: } \text{rel-preserves-binary-pred } \text{Rel } \text{Pred} \\
\text{shows } \text{rel-preserves-binary-pred } (\text{Rel}=) \text{ Pred} \\
\text{and } \text{rel-preserves-binary-pred } (\text{Rel}^+) \text{ Pred} \\
\text{and } \text{rel-preserves-binary-pred } (\text{Rel}^*) \text{ Pred}
\]

⟨ proof ⟩

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

lemma reflection-and-closures:

\[
\text{fixes } \text{Rel} :: \ ((a \times a) \text{ set}) \\
\text{and } \text{Pred} :: \ 'a \Rightarrow \text{Pred } 'b \Rightarrow \text{bool} \\
\text{assumes preservation: } \text{rel-preserves-binary-pred } \text{Rel } \text{Pred} \\
\text{shows } \text{rel-preserves-binary-pred } (\text{Rel}=) \text{ Pred} \\
\text{and } \text{rel-preserves-binary-pred } (\text{Rel}^+) \text{ Pred} \\
\text{and } \text{rel-preserves-binary-pred } (\text{Rel}^*) \text{ Pred}
\]

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

2 Process Calculi

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

2.1 Reduction Semantics

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

record 'proc processCalculus =
    Reductions :: 'proc ⇒ 'proc ⇒ bool
A pair of the reduction relation is called a (reduction) step.

**abbreviation** step :: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool

where

\[ P \rightarrow\text{Cal} Q \equiv \text{Reductions Cal P Q} \]

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

**primrec** nSteps

:: 'proc ⇒ 'proc processCalculus ⇒ nat ⇒ 'proc ⇒ bool

where

\[ P \rightarrow\text{Cal}^0 Q \equiv (P = Q) \]

\[ P \rightarrow\text{Cal}^{\text{Suc} n} Q = (\exists P'. P \rightarrow\text{Cal}^n P' \land P' \rightarrow\text{Cal} Q) \]

**definition** steps

:: 'proc ⇒ 'proc processCalculus ⇒ 'proc ⇒ bool

where

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

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

**definition** divergent

:: 'proc ⇒ 'proc processCalculus ⇒ bool

where

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

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

**lemma** steps-refl:

fixes Cal :: 'proc processCalculus

and P :: 'proc

shows P \rightarrow\text{Cal}^* P

(proof)

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

**lemma** step-to-steps:

fixes Cal :: 'proc processCalculus

and P :: 'proc

assumes step: P \rightarrow\text{Cal} P'

shows P \rightarrow\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:

fixes Cal :: 'proc processCalculus

and n1 n2 :: nat

shows \( \forall P Q R. P \rightarrow\text{Cal}^{n1} Q \land Q \rightarrow\text{Cal}^{n2} R \rightarrow P \rightarrow\text{Cal}^{(n1 + n2)} R \)

(proof)

**lemma** steps-add:

fixes Cal :: 'proc processCalculus

and P Q R :: 'proc

assumes A1: P \rightarrow\text{Cal}^* Q

and A2: Q \rightarrow\text{Cal}^* R

shows P \rightarrow\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.

```plaintext
record ('proc, 'barbs) calculusWithBarbs =
  Calculus :: 'proc processCalculus
  HasBarb :: 'proc ⇒ 'barbs ⇒ bool (¬<-[70, 70] 80)

abbreviation hasBarb :: 'proc ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs ⇒ bool
  where
  P↓<CWB>a ≡ HasBarb CWB P a
```

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

```plaintext
abbreviation reachesBarb :: 'proc ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs ⇒ bool
  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.

```plaintext
abbreviation rel-preserves-barb-set :: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs set ⇒ bool
  where
  rel-preserves-barb-set Rel CWB Barbs ≡ rel-preserves-binary-pred Rel (λP a. a ∈ Barbs ∧ P↓<CWB>a)
```

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

```plaintext
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.

```plaintext
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 (HasBarb CWB)
```

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

```plaintext
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.

```plaintext
abbreviation rel-respects-barb-set
```
\[
\text{rel-respects-barb-set Rel CWB Barbs} \equiv \\
\text{rel-preserves-barb-set Rel CWB Barbs} \land \text{rel-reflects-barb-set Rel CWB Barbs}
\]

\textbf{abbreviation rel-respects-barbs}
\[
\text{rel-respects-barbs} :: (\text{'proc } \times \text{'proc}) \text{ set} \rightarrow (\text{'proc, 'barbs}) \text{ calculusWithBarbs} \Rightarrow \text{'barbs set} \Rightarrow \text{bool}
\]
\[
\text{where}
\]
\[
\text{rel-respects-barbs Rel CWB Barbs} \equiv \\
\text{rel-preserves-barb-set Rel CWB Barbs} \land \text{rel-reflects-barb-set Rel CWB Barbs}
\]

\textbf{lemma respection-of-barbs-and-set-of-bars:}
\[
\text{fixes Rel} :: (\text{'proc } \times \text{'proc}) \text{ set} \\
\text{and CWB} :: (\text{'proc, 'barbs}) \text{ calculusWithBarbs}
\]
\[
\text{shows rel-respects-barbs Rel CWB} = (\forall \text{ Barbs. rel-respects-barb-set Rel CWB Barbs})
\]
\[
\langle \text{proof} \rangle
\]

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

\textbf{lemma preservation-of-barbs-and-closures:}
\[
\text{fixes Rel} :: (\text{'proc } \times \text{'proc}) \text{ set} \\
\text{and CWB} :: (\text{'proc, 'barbs}) \text{ calculusWithBarbs}
\]
\[
\text{assumes preservation: rel-preserves-barbs Rel CWB}
\]
\[
\text{shows rel-preserves-barbs (Rel}^\omega) \text{ CWB}
\]
\[
\text{and rel-preserves-barbs (Rel}^+ \text{) CWB}
\]
\[
\text{and rel-preserves-barbs (Rel}* \text{) CWB}
\]
\[
\langle \text{proof} \rangle
\]

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

\textbf{lemma reflection-of-barbs-and-closures:}
\[
\text{fixes Rel} :: (\text{'proc } \times \text{'proc}) \text{ set} \\
\text{and CWB} :: (\text{'proc, 'barbs}) \text{ calculusWithBarbs}
\]
\[
\text{assumes reflection: rel-reflects-barbs Rel CWB}
\]
\[
\text{shows rel-reflects-barbs (Rel}^\omega) \text{ CWB}
\]
\[
\text{and rel-reflects-barbs (Rel}^+ \text{) CWB}
\]
\[
\text{and rel-reflects-barbs (Rel}* \text{) CWB}
\]
\[
\langle \text{proof} \rangle
\]

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

\textbf{lemma respection-of-barbs-and-closures:}
\[
\text{fixes Rel} :: (\text{'proc } \times \text{'proc}) \text{ set} \\
\text{and CWB} :: (\text{'proc, 'barbs}) \text{ calculusWithBarbs}
\]
\[
\text{assumes respection: rel-respects-barbs Rel CWB}
\]
\[
\text{shows rel-respects-barbs (Rel}^\omega) \text{ CWB}
\]
\[
\text{and rel-respects-barbs (symcl Rel) CWB}
\]
\[
\text{and rel-respects-barbs (symcl (Rel}^\omega)) \text{ CWB}
\]
\[
\text{and rel-respects-barbs ((symcl (Rel}^\omega))^+ \text{) CWB}
\]
\[
\langle \text{proof} \rangle
\]

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.

\textbf{abbreviation rel-weakly-preserves-barb-set}
\[
\text{rel-weakly-preserves-barb-set} :: (\text{'proc } \times \text{'proc}) \text{ set} \Rightarrow (\text{'proc, 'barbs}) \text{ calculusWithBarbs} \Rightarrow \text{'barbs set} \Rightarrow \text{bool}
\]
\[
\text{where}
\]
\[
\text{rel-weakly-preserves-barb-set Rel CWB Barbs} \equiv \\
\text{rel-preserves-binary-pred Rel} (\lambda P. a \in \text{ Barbs} \land P \rho<\text{CWB}>a)
\]

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

**Lemma weak-preservation-of-barbs-and-set-of-barbs:**
- **fixes** $\text{Rel} :: (\text{proc} \times \text{proc})$ set
- and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs
- **shows** rel-weakly-preserves-barbs $\text{Rel} \ CWB$
  $$= (\forall \text{Barbs}. \ \text{rel-weakly-preserves-barb-set} \ \text{Rel} \ CWB \ \text{Barbs})$$

**Proof**

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
:: $(\text{proc} \times \text{proc})$ set $\Rightarrow$ $\text{rel-reflects-binary-pred} \ (\lambda P \ a. \ a \in \text{Barbs} \land P \Downarrow <\text{CWB} \ a)$

where

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

**Abbreviation** rel-weakly-preserves-barbs
:: $(\text{proc} \times \text{proc})$ set $\Rightarrow$ $(\text{proc}, \text{barbs})$ calculusWithBarbs $\Rightarrow$ $\text{barbs set} \Rightarrow$ bool

where

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

**Lemma weak-reflection-of-barbs-and-set-of-barbs:**
- **fixes** $\text{Rel} :: (\text{proc} \times \text{proc})$ set
- and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs
- **shows** rel-weakly-reflects-barbs $\text{Rel} \ CWB =$ $(\forall \text{Barbs}. \ \text{rel-weakly-reflects-barb-set} \ \text{Rel} \ CWB \ \text{Barbs})$

**Proof**

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

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

where

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

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

where

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

**Lemma weak-respection-of-barbs-and-set-of-barbs:**
- **fixes** $\text{Rel} :: (\text{proc} \times \text{proc})$ set
- and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs
- **shows** rel-weakly-respects-barbs $\text{Rel} \ CWB =$ $(\forall \text{Barbs}. \ \text{rel-weakly-respects-barb-set} \ \text{Rel} \ CWB \ \text{Barbs})$

**Proof**

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

**Lemma weak-preservation-of-barbs-and-closures:**
- **fixes** $\text{Rel} :: (\text{proc} \times \text{proc})$ set
- and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs
- **assumes** preservation: rel-weakly-preserves-barbs $\text{Rel} \ \text{CWB}$
- **shows** rel-weakly-preserves-barbs $\text{Rel}^= \ \text{CWB}$
  - rel-weakly-preserves-barbs $\text{Rel}^\uparrow \ \text{CWB}$
  - rel-weakly-preserves-barbs $\text{Rel}^\downarrow \ \text{CWB}$

**Proof**

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

**Lemma weak-reflection-of-barbs-and-closures:**
- **fixes** $\text{Rel} :: (\text{proc} \times \text{proc})$ set
If a relation weakly respects barbs then so does its reflexive, symmetric, or/and transitive closure.

\[ (\exists Q'. Q \mapsto Cal^* Q' \land (P', Q') \in Rel) \]

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

The reflexive and/or transitive closure of a weak simulation is a weak simulation.
lemma weak-barred-simulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: weak-barred-simulation Rel CWB
  shows weak-barred-simulation (Rel-) CWB
  and weak-barred-simulation (Rel+) CWB
  and weak-barred-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) ∈ Rel ∧ P ↓<CWB>a −→ Q ⇓<CWB>a

lemma weak-preservation-of-barbs:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes preservation: rel-weakly-preserves-barbs Rel CWB
  shows weak-barbed-preservation-cond Rel CWB
⟨proof⟩

lemma simulation-impl-equality-of-preservation-of-barbs-conditions:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: weak-reduction-simulation Rel CWB
  shows rel-weakly-preserves-barbs Rel CWB = 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) ∈ Rel ∧ P ↪−→ Cal P’ −→ (∃ Q’. Q ↪−→ Cal Q’ ∧ (P’, Q’) ∈ 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) ∧ 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:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes simulation: strong-barbed-simulation Rel CWB
shows weak-barbed-simulation Rel CWB
(proof)

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

Lemma strong-reduction-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes simulation: strong-reduction-simulation Rel Cal
shows strong-reduction-simulation (Rel") Cal
and strong-reduction-simulation (Rel+) Cal
and strong-reduction-simulation (Rel*) Cal
(proof)

Lemma strong-barbed-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes simulation: strong-barbed-simulation Rel CWB
shows strong-barbed-simulation Rel CWB
⟨proof⟩

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 × 'proc) set ⇒ 'proc processCalculus ⇒ bool
where
weak-reduction-contrasimulation Rel Cal ≡
∀ P Q P’. (P, Q) ∈ Rel ∧ P →∗ Cal P’ → (∃ Q’. Q →∗ Cal Q’ ∧ (Q’, P’) ∈ Rel)

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

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

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

Lemma weak-reduction-contrasimulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes contrasimulation: weak-reduction-contrasimulation Rel Cal
shows weak-reduction-contrasimulation (Rel") Cal
and weak-reduction-contrasimulation (Rel+) Cal
and weak-reduction-contrasimulation (Rel*) Cal
(proof)

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

⟨proof⟩

3.3 Coupled Simulation

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

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

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

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

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

lemma weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
shows weak-reduction-coupled-simulation Rel Cal
= (weak-reduction-simulation Rel Cal ∧ weak-reduction-contrasimulation Rel Cal)
⟨proof⟩

lemma weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
shows weak-barbed-coupled-simulation Rel CWB
= (weak-barbed-simulation Rel CWB ∧ weak-barbed-contrasimulation Rel CWB)
⟨proof⟩

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

lemma weak-reduction-coupled-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes coupledSimulation: weak-reduction-coupled-simulation Rel Cal
shows weak-reduction-coupled-simulation (Rel⁺) Cal
and weak-reduction-coupled-simulation (Rel*) Cal
⟨proof⟩

lemma weak-barbed-coupled-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes coupledSimulation: weak-barbed-coupled-simulation Rel CWB
shows weak-barbed-coupled-simulation (Rel⁺) CWB
and weak-barbed-coupled-simulation (Rel*) CWB
⟨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

$$\boxed{\langle \text{proc} \times \text{proc} \rangle \text{ set} \Rightarrow \text{proc processCalculus} \Rightarrow \text{bool}}$$

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

$$\boxed{\langle \text{proc} \times \text{proc} \rangle \text{ set} \Rightarrow \langle \text{proc, barbs} \rangle \text{ calculusWithBarbs} \Rightarrow \text{bool}}$$

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

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 :: \(\langle \text{proc} \times \text{proc} \rangle \text{ set} \Rightarrow \text{proc processCalculus} \Rightarrow \langle \text{proc} \times \text{proc} \rangle \text{ set}\)

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

$$\left\{ \begin{array}{l}
\text{fixes } \text{Rel} :: \langle \text{proc} \times \text{proc} \rangle \text{ set} \\
\text{and } \text{Cal} :: \langle \text{proc} \times \text{proc} \rangle \text{ set} \\
\text{where } \\
\#left: [Q \mapsto \text{Cal* } Q'; (P', Q') \in \text{Rel}] \Rightarrow (P', Q') \in \text{cSim-cs } \text{Rel Cal} |
\#right: [P \mapsto \text{Cal* } P'; (Q, P) \in \text{Rel}] \Rightarrow (P', Q) \in \text{cSim-cs } \text{Rel Cal} |
\#trans: [(P, Q) \in \text{cSim-cs } \text{Rel Cal}; (Q, R) \in \text{cSim-cs } \text{Rel Cal}] \Rightarrow (P, R) \in \text{cSim-cs } \text{Rel Cal} \\
\end{array} \right.$$
The reflexive and/or transitive closure of a weak correspondence simulation is a weak correspondence simulation.

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

- **fixes** \( \text{Rel} :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \)
- **and** \( \text{Cal} :: \text{\textquote{proc}} \text{\textquote{processCalculus}} \)
- **assumes** corrSim: weak-reduction-correspondence-simulation \( \text{Rel} \) \( \text{Cal} \)
- **shows** weak-reduction-correspondence-simulation \( (\text{Rel}^=) \) \( \text{Cal} \)
- **and** weak-reduction-correspondence-simulation \( (\text{Rel}^+) \) \( \text{Cal} \)
- **and** weak-reduction-correspondence-simulation \( (\text{Rel}^\ast) \) \( \text{Cal} \)

**proof**

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

- **fixes** \( \text{Rel} :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \)
- **and** \( \text{CWB} :: (\text{\textquote{proc}}, \text{\textquote{barbs}}) \text{\textquote{calculusWithBarbs}} \)
- **assumes** corrSim: weak-barbed-correspondence-simulation \( \text{Rel} \) \( \text{CWB} \)
- **shows** weak-barbed-correspondence-simulation \( (\text{Rel}^=) \) \( \text{CWB} \)
- **and** weak-barbed-correspondence-simulation \( (\text{Rel}^+) \) \( \text{CWB} \)
- **and** weak-barbed-correspondence-simulation \( (\text{Rel}^\ast) \) \( \text{CWB} \)

**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

- \( :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \Rightarrow \text{\textquote{proc}} \text{\textquote{processCalculus}} \Rightarrow \text{bool} \)
  - **where** weak-reduction-bisimulation \( \text{Rel} \) \( \text{Cal} \) \( \equiv \)
    - \( (\forall \ P Q P'. (P, Q) \in \text{Rel} \land P \xrightarrow{\text{\textquote{Cal}}} P' \implies (\exists Q'. Q \xrightarrow{\text{\textquote{Cal}}} Q' \land (P', Q') \in \text{Rel})) \)
    - \( (\forall \ P Q Q'. (P, Q) \in \text{Rel} \land Q \xrightarrow{\text{\textquote{Cal}}} Q' \implies (\exists P'. P \xrightarrow{\text{\textquote{Cal}}} P' \land (P', Q') \in \text{Rel})) \)

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

**abbreviation** weak-barbed-bisimulation

- \( :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \Rightarrow (\text{\textquote{proc}}, \text{\textquote{barbs}}) \text{\textquote{calculusWithBarbs}} \Rightarrow \text{bool} \)
  - **where** weak-barbed-bisimulation \( \text{Rel} \) \( \text{CWB} \) \( \equiv \)
    - weak-reduction-bisimulation \( \text{Rel} \) \( \text{CWB} \) \( \land \) rel-weakly-respects-barbs \( \text{Rel} \) \( \text{CWB} \)

A symmetric weak simulation is a weak bisimulation.

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

- **fixes** \( \text{Rel} :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \)
- **and** \( \text{Cal} :: \text{\textquote{proc}} \text{\textquote{processCalculus}} \)
- **assumes** sym \( \text{Rel} \)
  - **and** weak-reduction-simulation \( \text{Rel} \) \( \text{Cal} \)
- **shows** weak-reduction-bisimulation \( \text{Rel} \) \( \text{Cal} \)

**proof**

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

- **fixes** \( \text{Rel} :: (\text{\textquote{proc}} \times \text{\textquote{proc}}) \text{ set} \)
- **and** \( \text{CWB} :: (\text{\textquote{proc}}, \text{\textquote{barbs}}) \text{\textquote{calculusWithBarbs}} \)
- **assumes** sym \( \text{Rel} \)
  - **and** weak-barbed-simulation \( \text{Rel} \) \( \text{Cal} \)
- **shows** weak-barbed-bisimulation \( \text{Rel} \) \( \text{Cal} \)

**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-simulations-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⟩

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\(^{\omega}\)) Cal
  and weak-reduction-bisimulation (symcl Rel) Cal
  and weak-reduction-bisimulation (Rel\(^{+}\)) Cal
  and weak-reduction-bisimulation (symcl (Rel\(^{\omega}\))\(^{+}\)) 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\(^{\omega}\)) CWB
  and weak-barbed-bisimulation (symcl Rel) CWB
  and weak-barbed-bisimulation (Rel\(^{+}\)) CWB
  and weak-barbed-bisimulation (symcl (Rel\(^{\omega}\))\(^{+}\)) CWB
 ⟨proof⟩
\begin{itemize}
  \item \textbf{assumes} \textit{bisim}: weak-barbed-bisimulation \( \mathit{Rel} \mathit{CWB} \)
  \item \textbf{shows} weak-barbed-bisimulation \( (\mathit{Rel}^-) \mathit{CWB} \)
    \item and weak-barbed-bisimulation \( (\mathit{symcl} \mathit{Rel}) \mathit{CWB} \)
    \item and weak-barbed-bisimulation \( (\mathit{Rel}^+) \mathit{CWB} \)
    \item and weak-barbed-bisimulation \( (\mathit{symcl} (\mathit{Rel}^=)) \mathit{CWB} \)
    \item and weak-barbed-bisimulation \( (\mathit{symcl} (\mathit{Rel}^*)) \mathit{CWB} \)
    \item and weak-barbed-bisimulation \( ((\mathit{symcl} (\mathit{Rel}^=))^+) \mathit{CWB} \)
\end{itemize}

\begin{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 \((P', Q') \in R \).

\begin{itemize}
  \item \textbf{abbreviation} \textit{strong-reduction-bisimulation} \\
    \( :: (\text{'proc} \times \text{'proc}) \mathit{set} \Rightarrow \text{'proc processCalculus} \Rightarrow \text{bool} \)
    \item \textbf{where} \textit{strong-reduction-bisimulation} \( \mathit{Rel} \mathit{Cal} \equiv \) \\
    \((\forall P Q P'. (P, Q) \in \mathit{Rel} \land P \mapsto \mathit{Cal} P' \rightarrow (\exists Q'. Q \mapsto \mathit{Cal} Q' \land (P', Q') \in \mathit{Rel})) \) \\
    \land \((\forall P Q Q'. (P, Q) \in \mathit{Rel} \land Q \mapsto \mathit{Cal} Q' \rightarrow (\exists P'. P \mapsto \mathit{Cal} P' \land (P', Q') \in \mathit{Rel})) \)
\end{itemize}

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

\begin{itemize}
  \item \textbf{abbreviation} \textit{strong-barbed-bisimulation} \\
    \( :: (\text{'proc} \times \text{'proc}) \mathit{set} \Rightarrow (\text{'proc}, \text{'barbs}) \mathit{calculusWithBarbs} \Rightarrow \text{bool} \)
    \item \textbf{where} \textit{strong-barbed-bisimulation} \( \mathit{Rel} \mathit{CWB} \equiv \) \\
    \textit{strong-reduction-bisimulation} \( \mathit{Rel} \mathit{Cal} \land \mathit{rel-respects-barbs} \mathit{Rel} \mathit{CWB} \)
\end{itemize}

A symmetric strong simulation is a strong bisimulation.

\begin{lemma} \textbf{symm-strong-reduction-simulation-is-bisimulation}:
  \begin{itemize}
    \item \textbf{fixes} \( \mathit{Rel} :: (\text{'proc} \times \text{'proc}) \mathit{set} \)
    \item and \( \mathit{Cal} :: \text{'proc processCalculus} \)
    \item \textbf{assumes} \textit{sym} \( \mathit{Rel} \)
    \item and \textit{strong-reduction-simulation} \( \mathit{Rel} \mathit{Cal} \)
    \item \textbf{shows} \textit{strong-reduction-bisimulation} \( \mathit{Rel} \mathit{Cal} \)
  \end{itemize}
  \begin{proof}
\end{proof}
\end{lemma}

\begin{lemma} \textbf{symm-strong-barbed-simulation-is-bisimulation}:
  \begin{itemize}
    \item \textbf{fixes} \( \mathit{Rel} :: (\text{'proc} \times \text{'proc}) \mathit{set} \)
    \item and \( \mathit{CWB} :: (\text{'proc}, \text{'barbs}) \mathit{calculusWithBarbs} \)
    \item \textbf{assumes} \textit{sym} \( \mathit{Rel} \)
    \item and \textit{strong-barbed-simulation} \( \mathit{Rel} \mathit{CWB} \)
    \item \textbf{shows} \textit{strong-barbed-bisimulation} \( \mathit{Rel} \mathit{CWB} \)
  \end{itemize}
  \begin{proof}
\end{proof}
\end{lemma}

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

\begin{lemma} \textbf{strong-reduction-simulations-impl-bisimulation}:
  \begin{itemize}
    \item \textbf{fixes} \( \mathit{Rel} :: (\text{'proc} \times \text{'proc}) \mathit{set} \)
    \item and \( \mathit{Cal} :: \text{'proc processCalculus} \)
    \item \textbf{assumes} \textit{sim} \( \mathit{Rel} \mathit{Cal} \)
    \item and \textit{simInv} \( \mathit{Rel}^= \mathit{Cal} \)
    \item \textbf{shows} \textit{strong-reduction-bisimulation} \( \mathit{Rel}^= \mathit{Cal} \)
  \end{itemize}
  \begin{proof}
\end{proof}
\end{lemma}

\begin{lemma} \textbf{strong-reduction-bisimulations-impl-inverse-is-simulation}:
  \begin{itemize}
    \item \textbf{fixes} \( \mathit{Rel} :: (\text{'proc} \times \text{'proc}) \mathit{set} \)
    \item and \( \mathit{Cal} :: \text{'proc processCalculus} \)
    \item \textbf{assumes} \textit{bisim} \( \mathit{Rel} \mathit{Cal} \)
    \item \textbf{shows} \textit{strong-reduction-simulation} \( \mathit{Rel}^= \mathit{Cal} \)
  \end{itemize}
  \begin{proof}
\end{proof}
\end{lemma}
lemma strong-reduction-simulations-iff-bisimulation:
  fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
  shows (strong-reduction-simulation Rel Cal ∧ strong-reduction-simulation (Rel⁻¹) Cal)
         = strong-reduction-bisimulation Rel Cal
  ⟨proof⟩

lemma strong-barbed-simulations-iff-bisimulation:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  shows (strong-barbed-simulation Rel CWB ∧ strong-barbed-simulation (Rel⁻¹) CWB)
         = strong-barbed-bisimulation Rel CWB
  ⟨proof⟩

A strong bisimulation is a weak bisimulation.

lemma strong-impl-weak-reduction-bisimulation:
  fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
  assumes bisim: strong-reduction-bisimulation Rel Cal
  shows 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: strong-barbed-bisimulation Rel CWB
  shows 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: strong-barbed-bisimulation Rel CWB
  shows 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: strong-reduction-bisimulation Rel Cal
  shows strong-reduction-bisimulation (Rel⁺) Cal
       and strong-reduction-bisimulation (symcl Rel) Cal
       and strong-reduction-bisimulation (Rel⁺) Cal
       and strong-reduction-bisimulation (symcl (Rel⁺)) Cal
       and strong-reduction-bisimulation (symcl (Rel⁺)) Cal
       and 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: strong-barbed-bisimulation Rel CWB
  shows strong-barbed-bisimulation (Rel⁺) CWB
       and strong-barbed-bisimulation (symcl Rel) CWB
       and strong-barbed-bisimulation (Rel⁺) CWB
       and strong-barbed-bisimulation (symcl (Rel⁺)) CWB
       and strong-barbed-bisimulation (Rel⁺) CWB
       and strong-barbed-bisimulation ((symcl (Rel⁺))⁺) CWB
  ⟨proof⟩
3.6 Step Closure of Relations

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

**inductive-set** \( \text{stepsClosure} :: (a \times a) \text{ set} \Rightarrow a \text{ processCalculus} \Rightarrow (a \times a) \text{ set} \)

for Rel :: (a \times a) set

and Cal :: 'a processCalculus

where

rel: \((P, Q) \in \text{Rel} \implies (P, Q) \in \text{stepsClosure Rel Cal}\)

steps: \(P \mapsto \text{Cal} \Rightarrow P' \Rightarrow (P', P) \in \text{stepsClosure Rel Cal}\)

trans: \([P, Q] \in \text{stepsClosure Rel Cal}; (Q, R) \in \text{stepsClosure Rel Cal}\] \(\implies (P, R) \in \text{stepsClosure Rel Cal}\)

**abbreviation** \( \text{stepsClosureInfix} :: \)

\(a \Rightarrow (a \times a) \text{ set} \Rightarrow a \Rightarrow 'a \Rightarrow \text{bool} (-\text{R}\Rightarrow\text{<,> [-75, 75, 75, 75] 80})\)

where

\( P \text{ R}\Rightarrow\text{<Rel,Cal> Q} \equiv (P, Q) \in \text{stepsClosure Rel Cal}\)

Applying the steps closure twice does not change the relation.

**lemma** \( \text{steps-closure-of-steps-closure} :\)

fixes Rel :: (a \times a) set

and Cal :: 'a processCalculus

shows \(\text{stepsClosure} (\text{stepsClosure Rel Cal}) \text{ Cal} = \text{stepsClosure Rel Cal}\)

(proof)

The steps closure is a preorder.

**lemma** \( \text{stepsClosure-refl} :\)

fixes Rel :: (a \times a) set

and Cal :: 'a processCalculus

shows refl (stepsClosure Rel Cal)

(proof)

**lemma** \( \text{refl-trans-closure-of-rel-impl-steps-closure} :\)

fixes Rel :: (a \times a) set

and Cal :: 'a processCalculus

and P Q :: a

assumes \((P, Q) \in \text{Rel}^*\)

shows \(P \text{ R}\Rightarrow\text{<Rel,Cal> Q}\)

(proof)

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

**lemma** \( \text{steps-closure-is-weak-reduction-simulation} :\)

fixes Rel :: (a \times a) set

and Cal :: 'a processCalculus

shows weak-reduction-simulation (stepsClosure Rel Cal) Cal

(proof)

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

**lemma** \( \text{inverse-contrasimulation-impl-reverse-pair-in-steps-closure} :\)

fixes Rel :: (a \times a) set

and Cal :: 'a processCalculus

and P Q :: a

assumes con: weak-reduction-contrasimulation (Rel\(^{-1}\)) Cal

and pair: \((P, Q) \in \text{Rel}\)

shows \(Q \text{ R}\Rightarrow\text{<Rel,Cal> P}\)

(proof)

**lemma** \( \text{simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation} :\)
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:

```plaintext
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:

```plaintext
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:

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

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

```plaintext
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 P'.
(∃SP SP'. P = SourceTerm SP ∧ P' = SourceTerm SP' ∧ Reductions Source SP SP') ∨
(∃TP TP'. P = TargetTerm TP ∧ P' = TargetTerm TP' ∧ Reductions 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 ([]- [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)
A divergent term of Proc is either a divergent source term or a divergent target term.

abbreviation divergentST

A divergent term of Proc is either a divergent source term or a divergent target term.
lemma STCal-divergent:
  fixes S :: ′procS
  and T :: ′procT
  shows SourceTerm S ↦→ (STCal Source Target)ω = S ↦→ (Source)ω
  and TargetTerm T ↦→ (STCal Source Target)ω = T ↦→ (Target)ω
⟨proof⟩

lemma divergentST-STCal-divergent:
  fixes P :: (′procS, ′procT) Proc
  shows P ↦→ (STCal Source Target)ω = P ↦→ STω
⟨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(enc S) for all source terms S.

abbreviation enc-preserves-pred :: ((′procS, ′procT) Proc ⇒ bool) ⇒ bool where
  enc-preserves-pred Pred ≡ ∀S. Pred (SourceTerm S) → Pred (TargetTerm ([S]))

abbreviation enc-preserves-binary-pred :: ((′procS, ′procT) Proc ⇒ ′b ⇒ bool) ⇒ bool where
  enc-preserves-binary-pred Pred ≡ ∀S x. Pred (SourceTerm S) x → Pred (TargetTerm ([S])) x

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

abbreviation enc-reflects-pred :: ((′procS, ′procT) Proc ⇒ bool) ⇒ bool where
  enc-reflects-pred Pred ≡ ∀S. Pred (TargetTerm ([S])) → Pred (SourceTerm S)

abbreviation enc-reflects-binary-pred :: ((′procS, ′procT) Proc ⇒ ′b ⇒ bool) ⇒ bool where
  enc-reflects-binary-pred Pred ≡ ∀S x. Pred (TargetTerm ([S])) x → Pred (SourceTerm S) x

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

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

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

end

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

locale encoding-wrt-barbs =
  encoding Source Target Enc
  for Source :: ′procS processCalculus
  and Target :: ′procT processCalculus
  and Enc :: ′procS ⇒ ′procT +
  fixes SWB :: (′procS, ′barbs) calculusWithBarbs
  and TWB :: (′procT, ′barbs) calculusWithBarbs
  assumes calS: calculusWithBarbs, Calculus SWB = Source
  and calT: calculusWithBarbs, Calculus TWB = Target
begin
lemma $STCalWB$-$STCal$:
\[\text{shows Calculus} \ (STCalWB \ SWB \ TWB) = STCal \ Source \ Target\]
\[\langle \text{proof} \rangle\]

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

abbreviation $\text{hasBarbST}$
\[:: \ (\text{procS}'', \text{procT}) \ Proc \Rightarrow \ '\text{barbs} \Rightarrow bool \ (-\downarrow. [70, 70] 80)\]
where $P_{\downarrow. a} \equiv (\exists S. \ S \in S \ P \land S_{\downarrow. <SWB>a}) \lor (\exists T. \ T \in T \land T_{\downarrow. <TWB>a})$

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

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

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

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

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

abbreviation $\text{reachesBarbST}$
\[:: ('\text{procS}', '\text{procT}) \Proc \Rightarrow '\text{barbs} \Rightarrow bool \ (-\downarrow. [70, 70] 80)\]
where $P_{\downarrow. a} \equiv (\exists S. \ S \in S \ P \land S_{\downarrow. <SWB>a}) \lor (\exists T. \ T \in T \land T_{\downarrow. <TWB>a})$

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

Lemma weak-respection-of-barbs-in-barbed-encoding:

5 Relation between Source and Target Terms

5.1 Relations Induced by the Encoding Function

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

inductive-set (in encoding) indRelR
:: (((procS, procT) Proc × (procS, procT) Proc)) set
where
encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelR

abbreviation (in encoding) indRelRinfix ::
(procS, procT) Proc ⇒ (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
encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelRPO
source: (SourceTerm S, SourceTerm S) ∈ indRelRPO
Abbreviation (in encoding) indRelRPO-infiz ::
where
P \[ \leq \] R Q ≡ (P, Q) \in indRelRPO

Lemma (in encoding) indRelRPO-refl:
- shows refl indRelRPO
(proof)

Lemma (in encoding) indRelRPO-is-preorder:
- shows preorder indRelRPO
(proof)

Lemma (in encoding) refl-trans-closure-of-indRelR:
- shows indRelRPO = indRelR* (proof)

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

Lemma (in encoding) indRelR-impl-exists-source-target-relation:
- shows PredA indRelR ⇒ \exists Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) \in Rel) ∧ PredA Rel
  and \forall (P, Q) \in indRelR. PredB (P, Q)
  ⇒ \exists Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) \in Rel) ∧ (∀ (P, Q) \in Rel. PredB (P, Q))
(proof)

Lemma (in encoding) source-target-relation-impl-indRelR:
- assumes encRRel: ∀ S. (SourceTerm S, TargetTerm ([S])) \in Rel
  and condRel: ∀ (P, Q) \in Rel. Pred (P, Q)
- shows ∀ (P, Q) \in indRelR. Pred (P, Q)
(proof)

Lemma (in encoding) indRelR-iff-exists-source-target-relation:
- shows (∀ (P, Q) \in indRelR. Pred (P, Q))
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) \in Rel) ∧ (∀ (P, Q) \in Rel. Pred (P, Q)))
(proof)

Lemma (in encoding) indRelR-modulo-pred-impl-indRelRPO-modulo-pred:
- assumes reflCond: ∀ P. Pred (P, P)
  and transCond: ∀ P Q R. Pred (P, Q) ∧ Pred (Q, R) ⇒ Pred (P, R)
- shows (∀ (P, Q) \in indRelRPO. Pred (P, Q)) = (∀ (P, Q) \in indRelRPO. Pred (P, Q))
(proof)

Lemma (in encoding) indRelRPO-iff-exists-source-target-relation:
- shows (∀ (P, Q) \in indRelRPO. Pred (P, Q)) = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) \in Rel)
  ∧ (∀ (P, Q) \in Rel. Pred (P, Q)) ∧ preorder Rel)
(proof)

An encoding preserves, reflects, or respects a predicate iff indRelR preserves, reflects, or respects this predicate.
lemma \textbf{(in encoding)} \textbf{enc-satisfies-pred-impl-indRelR-satisfies-pred:}

\textbf{fixes} \textbf{Pred} :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool
\textbf{assumes} \textbf{encCond} : \forall S. \textbf{Pred} (\textbf{SourceTerm} S, \textbf{TargetTerm} ([S]))
\textbf{shows} \forall (P, Q) \in \textbf{indRelR}. \textbf{Pred} (P, Q)
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{indRelR-satisfies-pred-impl-enc-satisfies-pred:}

\textbf{fixes} \textbf{Pred} :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool
\textbf{assumes} \textbf{relCond} : \forall (P, Q) \in \textbf{indRelR}. \textbf{Pred} (P, Q)
\textbf{shows} \forall S. \textbf{Pred} (\textbf{SourceTerm} S, \textbf{TargetTerm} ([S]))
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred:}

\textbf{fixes} \textbf{Pred} :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool
\textbf{shows} \forall S \ a. \textbf{Pred} (\textbf{SourceTerm} S, \textbf{TargetTerm} ([S])) a = \forall (P, Q) \in \textbf{indRelR.} \forall a. \textbf{Pred} (P, Q) a
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-preserves-pred-iff-indRelR-preserves-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-preserves-pred Pred = rel-preserves-pred indRelR Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-preserves-binary-pred-iff-indRelR-preserves-binary-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-preserves-binary-pred Pred = rel-preserves-binary-pred indRelR Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-preserves-pred-iff-indRelRPO-preserves-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-preserves-pred Pred = rel-preserves-pred indRelRPO Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-reflects-pred-iff-indRelR-reflects-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-reflects-pred Pred = rel-reflects-pred indRelR Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-reflects-binary-pred Pred = rel-reflects-binary-pred indRelR Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-reflects-pred-iff-indRelRPO-reflects-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-reflects-pred Pred = rel-reflects-pred indRelRPO Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-respects-pred-iff-indRelR-respects-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-respects-pred Pred = rel-respects-pred indRelR Pred}
\langle proof \rangle

lemma \textbf{(in encoding)} \textbf{enc-respects-binary-pred-iff-indRelR-respects-binary-pred:}

\textbf{fixes} \textbf{Pred} :: ('procS, 'procT) Proc \Rightarrow bool
\textbf{shows} \textbf{enc-respects-binary-pred Pred = rel-respects-binary-pred indRelR Pred}
\langle proof \rangle
for all source terms S.

To analyse the reflection of source term behaviours we use relations that contain the pairs (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.

and enc-respects-pred TargetTerm (P, Q) ⇒ Pred (P, Q); ∀ P. Pred (P, P) ⇒
(∀ S. Pred (SourceTerm S, TargetTerm (⟦S⟧))) = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) ∧ (∀ P, Q) ∈ Rel. Pred (P, Q))

and enc-preserves-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel)
∧ rel-preserves-pred Rel Pred ∧ preorder Rel)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-pred Pred) ⇒ false

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-pred Rel Pred ∧ preorder Rel)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-preserves-binary-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)

∧ rel-respects-pred Rel Pred)
inductive-set (in encoding) indRelL
:: (((procS, procT) Proc) × ((procS, procT) Proc)) set
where
encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelL

abbreviation (in encoding) indRelLinfix ::
where
P R L Q ≡ (P, Q) ∈ indRelL

inductive-set (in encoding) indRelLPO
:: (((procS, procT) Proc) × ((procS, procT) Proc)) set
where
encL: (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 \( \text{rel-respects-pred} \; \text{indRelR} \; \text{Pred} = \text{rel-respects-pred} \; \text{indRelL} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-respects-binary-pred-iff-indRelL-respects-binary-pred} \):

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

shows \( \text{rel-respects-binary-pred} \; \text{indRelR} \; \text{Pred} = \text{rel-respects-binary-pred} \; \text{indRelL} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-cond-preservation-iff-indRelL-cond-reflection} \):

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

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

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

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-cond-binary-preservation-iff-indRelL-cond-binary-reflection} \):

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

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

\( \land \text{rel-reflects-binary-pred} \; \text{Rel} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-cond-reflection-iff-indRelL-cond-preservation} \):

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

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

\( \land \text{rel-preserves-binary-pred} \; \text{Rel} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-cond-binary-reflection-iff-indRelL-cond-binary-preservation} \):

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

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

\( \land \text{rel-preserves-binary-pred} \; \text{Rel} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{indRelL-cond-respection-iff-indRelL-cond-respection} \):

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

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

\( \land \text{rel-respects-binary-pred} \; \text{Rel} \; \text{Pred} \)

(\text{proof})

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

**Lemma (in encoding)** \( \text{enc-preserves-pred-iff-indRelL-reflects-pred} \):

\( \text{fixes} \; \text{Pred} :: (\text{procS}, \text{procT}) \; \text{Proc} \Rightarrow \text{bool} \)

shows \( \text{enc-preserves-pred} \; \text{Pred} = \text{rel-reflects-pred} \; \text{indRelL} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{enc-reflects-pred-iff-indRelL-preserves-pred} \):

\( \text{fixes} \; \text{Pred} :: (\text{procS}, \text{procT}) \; \text{Proc} \Rightarrow \text{bool} \)

shows \( \text{enc-reflects-pred} \; \text{Pred} = \text{rel-preserves-pred} \; \text{indRelL} \; \text{Pred} \)

(\text{proof})

**Lemma (in encoding)** \( \text{enc-respects-pred-iff-indRelL-respects-pred} \):
An encoding preserves, reflects, or respects a predicate iff there exists a relation, namely indRelL, that relates literal translations with their source terms and reflects, preserves, or respects this predicate.

**Lemma (in encoding)** enc-preserves-pred-iff-source-target-rel-preserves-pred:

- **Fixes*** $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$
- **Show** $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred})$

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)** $\text{indRel}$

- $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$
- **Where***
  - $\text{encR}: (\text{SourceTerm } S, \text{TargetTerm } [S]) \in \text{indRel}$
  - $\text{encL}: (\text{TargetTerm } [S], \text{SourceTerm } S) \in \text{indRel}$

**Abbreviation (in encoding)** $\text{indRelInfix}$:

- $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$
- **Where***
  - $P \in \text{indRel}[P, Q]$ $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$

**Lemma (in encoding)** $\text{indRel-symm}$:

- **Show** $\text{sym } \text{indRel}$

**Inductive-set (in encoding)** $\text{indRelEQ}$

- $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$
- **Where***
  - $\text{encR}: (\text{SourceTerm } S, \text{TargetTerm } [S]) \in \text{indRelEQ}$
  - $\text{encL}: (\text{TargetTerm } [S], \text{SourceTerm } S) \in \text{indRelEQ}$
  - $\text{target}: (\text{TargetTerm } T, \text{TargetTerm } T) \in \text{indRelEQ}$
  - $\text{trans}: [(P, Q) \in \text{indRelEQ}; (Q, R) \in \text{indRelEQ}] \rightarrow (P, R) \in \text{indRelEQ}$

**Abbreviation (in encoding)** $\text{indRelEQInfix}$:

- $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$
- **Where***
  - $P \in \text{indRelEQ}[P, Q]$ $\forall S. (\text{SourceTerm } S \in \text{Rel}) \land \text{rel-preserves-pred } \text{Rel } \text{Pred}$

**Lemma (in encoding)** $\text{indRelEQ-refl}$:

- **Show** $\text{refl } \text{indRelEQ}$

**Lemma (in encoding)** $\text{indRelEQ-is-preorder}$:

- **Show** $\text{preorder } \text{indRelEQ}$
lemma (in encoding) indRelEQ-symm:
  shows symm 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 (indRelR^=))^+
(proof)

lemma (in encoding) symm-closure-of-indRelL:
  shows indRel = symcl indRelL
  and indRelEQ = (symcl (indRelL^=))^+
(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 (3 Rel.
      (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
      ∧ rel-respects-pred Rel Pred)
      = (3 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 (3 Rel.
      (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
      ∧ rel-respects-binary-pred Rel Pred)
      = (3 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 \( \text{enc-respects-pred \ Pred} = \text{rel-respects-pred \ indRel \ Pred} \)

(proof)

An encoding respects a predicate iff there exists a relation, namely \( \text{indRel} \), that relates source terms and their literal translations in both directions and respects this predicate.

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

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

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

(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) \( \text{indRelRT} \)
\[ :: (\text{procT} \times \text{procT}) \text{ set} \Rightarrow (((\text{procS}, \text{procT}) \text{ Proc}) \times ((\text{procS}, \text{procT}) \text{ Proc})) \text{ set} \]

for \( \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \)

where
\[ \text{encR} :: (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{indRelRT \ TRel |} \]

\[ \text{target} :: (T1, T2) \in \text{TRel} \Rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{indRelRT \ TRel} \]

abbreviation (in encoding) \( \text{indRelRTinfix} \)
\[ :: (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow (\text{procT} \times \text{procT}) \text{ set} \Rightarrow (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow \text{bool} \]
\[ (- \cdot \ \R[\cdot]RT<\cdot> - [75, 75, 75] 80) \]

where
\[ P \ \R[\cdot]RT<\cdot> Q \equiv (P, Q) \in \text{indRelRT \ TRel} \]

inductive-set (in encoding) \( \text{indRelRTPO} \)
\[ :: (\text{procT} \times \text{procT}) \text{ set} \Rightarrow (((\text{procS}, \text{procT}) \text{ Proc}) \times ((\text{procS}, \text{procT}) \text{ Proc})) \text{ set} \]

for \( \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \)

where
\[ \text{encR} :: (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{indRelRTPO \ TRel |} \]

\[ \text{source} :: (\text{SourceTerm} S, \text{SourceTerm} S) \in \text{indRelRTPO \ TRel |} \]

\[ \text{target} :: (T1, T2) \in \text{TRel} \Rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{indRelRTPO \ TRel |} \]

\[ \text{trans} :: ([P, Q]) \in \text{indRelRTPO \ TRel} \Rightarrow (Q, R) \in \text{indRelRTPO \ TRel} \Rightarrow (P, R) \in \text{indRelRTPO \ TRel} \]

abbreviation (in encoding) \( \text{indRelRTPOinfix} \)
\[ :: (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow (\text{procT} \times \text{procT}) \text{ set} \Rightarrow (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow \text{bool} \]
\[ (- \cdot \ \R[\cdot]RT<\cdot> - [75, 75, 75] 80) \]

where
\[ P \ \R[\cdot]RT<\cdot> Q \equiv (P, Q) \in \text{indRelRTPO \ TRel} \]

lemma (in encoding) \( \text{indRelRTPO-refl} \):

fixes \( \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \)

assumes \( \text{refl \ TRel} \)

shows \( \text{refl \ (indRelRTPO \ TRel)} \)

(proof)

lemma (in encoding) \( \text{refl-trans-closure-of-indRelRT} \):

fixes \( \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \)

assumes \( \text{refl \ TRel} \)

shows \( \text{indRelRTPO \ TRel} = (\text{indRelRT \ TRel})^* \)

(proof)

lemma (in encoding) \( \text{indRelRTPO-is-preorder} \):
The relation $\text{indRelLT}$ includes $\text{TRel}$ and relates literal translations and their source terms.

**Lemma (in encoding)** $\text{transitive-closure-of-$\text{TRel}$-to-$\text{indRelRTPO}$}:

- **Fixes** $\text{TRel :: ('}\text{procT} \times '\text{procT}'\text{) set}$
- **Assumes** $\text{reflT :: refl TRel}$
- **Shows** $\text{preorder (}$ $\text{indRelRTPO}$ $\text{TRel}$ $\text{)}$

\[ \text{proof} \]

The relation $\text{indRelRT}$ is the smallest relation that relates all source terms and their literal translations and contains $\text{TRel}$. Thus there exists a relation that relates source terms and their literal translations and satisfies some predicate on its pairs iff the predicate holds for the pairs of $\text{indRelR}$.

**Lemma (in encoding)** $\text{indRelRT-modulo-pred-implied-indRelRT-modulo-pred}:

- **Fixes** $\text{Pred :: ('}\text{procS} , '\text{procT}'\text{) Proc } \times ('}\text{procS} , '\text{procT}'\text{) Proc} \Rightarrow bool$
- **Shows** $\forall (P , Q) \in \text{indRelR} \text{.} \text{Pred (P , Q)} = (\forall (TP , TQ) \in \text{TRel} \text{.} \text{Pred (TargetTerm TP , TargetTerm TQ))}$

\[ \text{proof} \]

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

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

- **:: (}$ '\text{procT} \times '\text{procT}'\text{) set$}
- **For** $\text{TRel :: ('}\text{procT} \times '\text{procT}'\text{) set}$
- **Where** $\text{encL :: ('}\text{procS} , '\text{procT}''\text{) Proc} = ('}\text{procS} , '\text{procT}''\text{) Proc} \Rightarrow bool$

\[ \text{where} \]

**Abbreviation (in encoding)** $\text{indRelLTinfix}:

- $\text{:: (}$ '\text{procS} , '\text{procT}'\text{) Proc} \Rightarrow ('}\text{procT} \times '\text{procT}'\text{) set} \Rightarrow ('}\text{procS} , '\text{procT}''\text{) Proc} \Rightarrow bool$

\[ \text{where} \]

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

- $\text{:: (}$ '\text{procT} \times '\text{procT}'\text{) set} \Rightarrow (}$ '\text{procS} , '\text{procT}'\text{) Proc$} \times (}$ '\text{procS} , '\text{procT}'\text{) Proc$} \Rightarrow bool$
lemma \((\text{encoding})\) \(\text{indRelLTPoinfix}\)
\[\begin{align*}
\text{abbreviation} & \text{ (encoding) } \text{indRelLTPoinfix} \\
& \quad \text{:: } (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow (\text{procT } 	imes \text{procT}) \text{ set } \Rightarrow (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow bool \\
& \quad (- \lesssim [\text{LT}]_{<>} - [75, 75, 75] 80) \\
\text{where} & \\
&P \lesssim [\text{LT}]_{<>} Q \equiv (P, Q) \in \text{indRelLTPo TRel}
\end{align*}\]

lemma \((\text{encoding})\) \(\text{indRelLTPo-refl}\):
\[\begin{align*}
\text{fixes} & \text{ TRel } \text{:: } (\text{procT } 	imes \text{procT}) \text{ set } \\
\text{assumes} & \text{ refl: refl TRel} \\
\text{shows} & \text{ refl (indRelLTPo TRel)} \\
\text{(proof)} &
\end{align*}\]

lemma \((\text{encoding})\) \(\text{refl-trans-closure-of-indRelLT}\):
\[\begin{align*}
\text{fixes} & \text{ TRel } \text{:: } (\text{procT } 	imes \text{procT}) \text{ set } \\
\text{assumes} & \text{ refl: refl TRel} \\
\text{shows} & \text{ indRelLTPo TRel } = (\text{indRelLT TRel})^* \\
\text{(proof)} &
\end{align*}\]

inductive-set \((\text{encoding})\) \(\text{indRelT}\)
\[\begin{align*}
\text{abbreviation} & \text{ (encoding) } \text{indRelTinfix} \\
& \quad \text{:: } (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow (\text{procT } 	imes \text{procT}) \text{ set } \Rightarrow (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow bool \\
& \quad (- \lesssim [\text{T}]_{<>} - [75, 75, 75] 80) \\
\text{where} & \\
&P \lesssim [\text{T}]_{<>} Q \equiv (P, Q) \in \text{indRelT TRel}
\end{align*}\]

lemma \((\text{encoding})\) \(\text{indRelT-symm}\):
\[\begin{align*}
\text{fixes} & \text{ TRel } \text{:: } (\text{procT } 	imes \text{procT}) \text{ set } \\
\text{assumes} & \text{ symm: sym TRel} \\
\text{shows} & \text{ symm (indRelT TRel)} \\
\text{(proof)} &
\end{align*}\]

inductive-set \((\text{encoding})\) \(\text{indRelTEQ}\)
\[\begin{align*}
\text{abbreviation} & \text{ (encoding) } \text{indRelTEQinfix} \\
& \quad \text{:: } (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow (\text{procT } 	imes \text{procT}) \text{ set } \Rightarrow (\text{procS}, \text{procT}) \text{ Proc } \Rightarrow bool \\
& \quad (- \lesssim [\text{T}]_{<>} - [75, 75, 75] 80) \\
\text{where} & \\
&P \lesssim [\text{T}]_{<>} Q \equiv (P, Q) \in \text{indRelTEQ TRel}
\end{align*}\]

lemma \((\text{encoding})\) \(\text{indRelTEQ-refl}\):
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⟩

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 \[\cdot\]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 \[\cdot\]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 \[\cdot\]T<TRel TargetTerm TQ

\[
\text{shows } (TP, TQ) \in TRel
\]

\{proof\}

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

**Lemma (in encoding) indRelRTPO-to-TRel:**

- **fixes** \(\text{TRel} : (\text{\textquotesingleprocT} \times \text{\textquotesingleprocT}) \text{set}\)
- **and** \(P Q : (\text{\textquotesingleprocS}, \text{\textquotesingleprocT}) \text{Proc}\)
- **assumes** \(\text{rel} : P \preceq RT < TRel > Q\)
- **shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow SP = SQ\)
- **shows** \(\forall SP TQ. SP \in S P \land TQ \in T Q \rightarrow (\{SP\}, TQ) \in (TRel \cup \{(T1, T2)\}. \exists S. T1 \in [S] \land T2 \in [S]\)) +\)
- **shows** \(\forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow False\)
- **shows** \(\forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in TRel +\)

\{proof\}

**Lemma (in encoding) indRelLTPO-to-TRel:**

- **fixes** \(\text{TRel} : (\text{\textquotesingleprocT} \times \text{\textquotesingleprocT}) \text{set}\)
- **and** \(P Q : (\text{\textquotesingleprocS}, \text{\textquotesingleprocT}) \text{Proc}\)
- **assumes** \(\text{rel} : P \succeq LT < TRel > Q\)
- **shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow SP = SQ\)
- **shows** \(\forall SP TP SQ. SP \in S P \land TP \in T Q \land SQ \in S Q \rightarrow False\)
- **shows** \(\forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in TRel +\)

\{proof\}

**Lemma (in encoding) indRelTEQ-to-TRel:**

- **fixes** \(\text{TRel} : (\text{\textquotesingleprocT} \times \text{\textquotesingleprocT}) \text{set}\)
- **and** \(P Q : (\text{\textquotesingleprocS}, \text{\textquotesingleprocT}) \text{Proc}\)
- **assumes** \(\text{rel} : P \sim T < TRel > Q\)
- **shows** \(\forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow (\{SP\}, \{SQ\}) \in (TRel \cup \{(T1, T2)\}. \exists S. T1 \in [S] \land T2 \in [S]\) +\)
- **shows** \(\forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow (TP, SQ) \in (TRel \cup \{(T1, T2)\}. \exists S. T1 \in [S] \land T2 \in [S]\) +\)
- **shows** \(\forall TP TQ. TP \in T P \land TQ \in T T Q \rightarrow (TP, TQ) \in TRel *\)

\{proof\}

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

**Lemma (in encoding) indRelRTPO-relates-source-target:**

- **fixes** \(\text{TRel} : (\text{\textquotesingleprocT} \times \text{\textquotesingleprocT}) \text{set}\)
- **and** \(P Q : (\text{\textquotesingleprocT}, \text{\textquotesingleprocT}) \text{Proc}\)
- **assumes** \(\text{pair : SourceTerm S} \preceq RT \text{< TRel > TargetTerm T}\)
- **shows** \(\text{(TargetTerm ([S]), TargetTerm T)} \in (\text{indRelRTPO TRel})^\ast\)

\{proof\}

If \(\text{indRelRTPO}\), \(\text{indRelLTPO}\), or \(\text{indRelTPO}\) preserves barbs then so does the corresponding target 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.

lemma (in encoding-wrt-barbs) rel-with-target-impl-TRel-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)
  and targetInRel: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
shows rel-reflects-barbs TRel TWB
(proof)

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)

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

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

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

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) 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⟩

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 \((TRel^+)\) Target

(proof)

**lemma (in encoding)** indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev:

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

assumes sim: weak-reduction-simulation \((\text{indRelLTPO } TRel)^{-1}\) (STCal Source Target)

shows weak-reduction-simulation \((TRel^+)\) Target

(proof)

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

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

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

assumes sim: weak-reduction-simulation \(Rel\) (STCal Source Target)

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

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

shows weak-reduction-simulation \((TRel^+)\) Target

(proof)

**lemma (in encoding)** indRelTEQ-impl-TRel-is-weak-reduction-simulation:

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

assumes sim: weak-reduction-simulation \((\text{indRelTEQ } TRel)\) (STCal Source Target)

shows weak-reduction-simulation \((TRel^+)\) Target

(proof)

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

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

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

assumes sim: weak-reduction-simulation \(Rel\) (STCal Source Target)

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

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

shows weak-reduction-simulation \((TRel^+)\) Target

(proof)

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

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

assumes sim: strong-reduction-simulation \((\text{indRelRTPO } TRel)\) (STCal Source Target)

shows strong-reduction-simulation \((TRel^+)\) Target

(proof)

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

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

assumes sim: strong-reduction-simulation \((\text{indRelLTPO } TRel)\) (STCal Source Target)

shows strong-reduction-simulation \((TRel^+)\) Target

(proof)

**lemma (in encoding)** rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev:

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

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

assumes sim: strong-reduction-simulation \(Rel\) (STCal Source Target)

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

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

shows strong-reduction-simulation \((TRel^+)^{-1}\) Target

(proof)

**lemma (in encoding)** indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev:

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

assumes sim: strong-reduction-simulation \((\text{indRelRTPO } TRel)^{-1}\) (STCal Source Target)

shows strong-reduction-simulation \((TRel^+)^{-1}\) Target

(proof)

**lemma (in encoding)** indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev:


lemma (in encoding) rel-with-target-impl-refC-transC-TRel-is-strong-reduction-simulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes sim: strong-reduction-simulation (Rel (STCal Source Target))
  shows strong-reduction-simulation (TRel+) 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-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-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)

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

and \(\text{Rel} \:: \langle \langle \text{procS}, \text{procT} \rangle \rangle \) set

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

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

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

shows \(\text{weak-reduction-contrasimulation} \ (TRel^+) \) Target

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-reduction-contrasimulation} \ (TRel^+) \) Target

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

and \(\text{Rel} \:: \langle \langle \text{procS}, \text{procT} \rangle \rangle \) set

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

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

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

shows \(\text{weak-reduction-contrasimulation} \ (TRel^+) \) Target

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-reduction-contrasimulation} \ (TRel^+) \) Target

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-barbed-contrasimulation} \ (TRel^+) \) \(\text{STCal WB SWB TWB}\)

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-barbed-contrasimulation} \ (TRel^+) \) \(\text{STCal WB SWB TWB}\)

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

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

fixes \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-barbed-contrasimulation} \ (TRel^+) \) \(\text{STCal WB SWB TWB}\)

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

If \(\text{indRelRTPO}\), \(\text{indRelLTPO}\), or \(\text{indRelTEQ}\) 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 \(TRel \:: \langle \text{procT} \times \text{procT} \rangle \) set

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

shows \(\text{weak-reduction-coupled-simulation} \ (TRel^+) \) Target

\(\langle \text{proof} \rangle\)
\textbf{lemma} (in encoding) \texttt{indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{couSim: weak-reduction-coupled-simulation (indRelLTPO TRel) (STCal Source Target)}
  \item \texttt{shows} \texttt{weak-reduction-coupled-simulation (TRel+) Target}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelTEQ-impl-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{couSim: weak-reduction-coupled-simulation (indRelTEQ TRel) (STCal Source Target)}
  \item \texttt{shows} \texttt{weak-reduction-coupled-simulation (TRel+) Target}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelRTPO-impl-TRel-is-weak-barbed-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{couSim: weak-barbed-coupled-simulation (indRelRTPO TRel) (STCalWB SWB TWB)}
  \item \texttt{shows} \texttt{weak-barbed-coupled-simulation (TRel+) TWB}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelLTPO-impl-TRel-is-weak-barbed-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{couSim: weak-barbed-coupled-simulation (indRelLTPO TRel) (STCalWB SWB TWB)}
  \item \texttt{shows} \texttt{weak-barbed-coupled-simulation (TRel+) TWB}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelTEQ-impl-TRel-is-weak-barbed-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{couSim: weak-barbed-coupled-simulation (indRelTEQ TRel) (STCalWB SWB TWB)}
  \item \texttt{shows} \texttt{weak-barbed-coupled-simulation (TRel+) TWB}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{rel-with-target-impl-transC-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{and} \texttt{Rel :: \langle \langle\langle procS, \langle procT \rangle \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{corSim: weak-reduction-coupled-simulation Rel (STCal Source Target)}
  \item \texttt{and} \texttt{target: \forall T1 T2. (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel}
  \item \texttt{and} \texttt{trel: \forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel+}
  \item \texttt{shows} \texttt{weak-reduction-coupled-simulation (TRel+) Target}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelRTPO-impl-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{cSim: weak-reduction-coupled-simulation (indRelRTPO TRel) (STCal Source Target)}
  \item \texttt{shows} \texttt{weak-reduction-coupled-simulation (TRel+) Target}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{cSim: weak-reduction-coupled-simulation (indRelLTPO TRel) (STCal Source Target)}
  \item \texttt{shows} \texttt{weak-reduction-coupled-simulation (TRel+) Target}
\end{itemize}
\begin{proof}
\end{proof}

\textbf{lemma} (in encoding) \texttt{rel-with-target-impl-refC-TRel-is-weak-reduction-coupled-simulation}:
\begin{itemize}
  \item \texttt{fixes} \texttt{TRel :: \langle \langle procT \times \langle procT \rangle \rangle \rangle \set}
  \item \texttt{and} \texttt{Rel :: \langle \langle\langle procS, \langle procT \rangle \rangle \rangle \rangle \set}
  \item \texttt{assumes} \texttt{corSim: weak-reduction-coupled-simulation Rel (STCal Source Target)}
  \item \texttt{and} \texttt{target: \forall T1 T2. (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel}
\end{itemize}
\begin{proof}
\end{proof}
and \( \text{trel}: \forall T1 T2. \) \((\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \quad \rightarrow \quad (T1, T2) \in \text{TRel}^+\)

shows weak-reduction-correspondence-simulation \((\text{TRel}^+)\) Target

(proof)

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

fixes \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{ set}\)

assumes \(\text{corSim}: \) weak-reduction-correspondence-simulation \((\text{indRelTEQ TRel}) (\text{STCal Source Target})\)

shows weak-reduction-correspondence-simulation \((\text{TRel}^+)\) TWB

(proof)

If \(\text{indRelRTPO}, \text{indRelLTPO},\) or \(\text{indRelTEQ}\) 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 \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{ set}\)

and \(\text{Rel} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \text{ set}\)

assumes \(\text{bism}: \text{ weak-reduction-bisimulation Rel} (\text{STCal Source Target})\)

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

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

shows weak-reduction-bisimulation \((\text{TRel}^+)\) Target

(proof)

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

fixes \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{ set}\)

assumes \(\text{bism}: \) weak-reduction-bisimulation \((\text{indRelRTPO TRel}) (\text{STCalWB SWB TWB})\)

shows weak-barbed-correspondence-simulation \((\text{TRel}^+)\) TWB

(proof)

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

fixes \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{ set}\)

assumes \(\text{bism}: \) weak-reduction-bisimulation \((\text{indRelLTPO TRel}) (\text{STCalWB SWB TWB})\)

shows weak-barbed-correspondence-simulation \((\text{TRel}^+)\) TWB

(proof)

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

fixes \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{ set}\)

assumes \(\text{bism}: \) weak-reduction-bisimulation \((\text{indRelTEQ TRel}) (\text{STCalWB SWB TWB})\)

shows weak-barbed-correspondence-simulation \((\text{TRel}^+)\) TWB

(proof)
lemma (in encoding) \( \text{indRelTEQ-impl-TRel-is-weak-reduction-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{weak-reduction-bisimulation} \ (\text{indRelTEQ TRel}) \) \( \text{(STCal Source Target)} \)
  \( \text{shows } \text{weak-reduction-bisimulation} \ (TRel^+) \) \( \text{Target} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{and } \text{Rel} :: ((\text{proc} S, \text{proc} T) \text{Proc} \times (\text{proc} S, \text{proc} T) \text{Proc}) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{strong-reduction-bisimulation} \ (\text{Rel} \text{(STCal Source Target)}) \)
  \( \text{and } \text{target}: \forall T_1 T_2. (T_1, T_2) \in TRel \rightarrow ((\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \text{and trl}) \)
  \( \forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in TRel^+ \)
  \( \text{shows } \text{strong-reduction-bisimulation} \ (TRel^+) \) \( \text{Target} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelRTPO-impl-TRel-is-strong-reduction-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{strong-reduction-bisimulation} \ (\text{indRelRTPO TRel}) \) \( \text{(STCal Source Target)} \)
  \( \text{shows } \text{strong-reduction-bisimulation} \ (TRel^+) \) \( \text{Target} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelLTPO-impl-TRel-is-strong-reduction-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{strong-reduction-bisimulation} \ (\text{indRelLTPO TRel}) \) \( \text{(STCal Source Target)} \)
  \( \text{shows } \text{strong-reduction-bisimulation} \ (TRel^+) \) \( \text{Target} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{and } \text{Rel} :: ((\text{proc} S, \text{proc} T) \text{Proc} \times (\text{proc} S, \text{proc} T) \text{Proc}) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{strong-reduction-bisimulation} \ (\text{Rel} \text{(STCal Source Target)}) \)
  \( \text{and } \text{target}: \forall T_1 T_2. (T_1, T_2) \in TRel \rightarrow ((\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \text{and trl}) \)
  \( \forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in TRel^+ \)
  \( \text{shows } \text{strong-reduction-bisimulation} \ (TRel^+) \) \( \text{Target} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelTEQ-impl-TRel-is-weak-barbed-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{weak-barbed-bisimulation} \ (\text{indRelTEQ TRel}) \) \( \text{(STCal Source Target)} \)
  \( \text{shows } \text{weak-barbed-bisimulation} \ (TRel^+) \) \( \text{TWB} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelRTPO-impl-TRel-is-weak-barbed-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{weak-barbed-bisimulation} \ (\text{indRelRTPO TRel}) \) \( \text{(STCal WB SWB TWB)} \)
  \( \text{shows } \text{weak-barbed-bisimulation} \ (TRel^+) \) \( \text{TWB} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelLTPO-impl-TRel-is-weak-barbed-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{weak-barbed-bisimulation} \ (\text{indRelLTPO TRel}) \) \( \text{(STCal WB SWB TWB)} \)
  \( \text{shows } \text{weak-barbed-bisimulation} \ (TRel^+) \) \( \text{TWB} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelTEQ-impl-TRel-is-weak-barbed-bisimulation} \):
  \( \text{fixes } TRel :: (\text{proc} T \times \text{proc} T) \) \( \text{set} \)
  \( \text{assumes } \text{bisim}: \text{weak-barbed-bisimulation} \ (\text{indRelTEQ TRel}) \) \( \text{(STCal WB SWB TWB)} \)
  \( \text{shows } \text{weak-barbed-bisimulation} \ (TRel^+) \) \( \text{TWB} \)
  \( \langle \text{proof} \rangle \)

lemma (in encoding) \( \text{indRelRTPO-impl-TRel-is-strong-barbed-bisimulation} \):
lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-strong-barbed-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: strong-barbed-bisimulation (indRelLTPO TRel) (STCalWB SWB TWB)
  shows strong-barbed-bisimulation (TRel⁺) TWB
  (proof)

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
  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) indRelRSTO
  :: ('procS, 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
  where
  P ∊ {\{R ≲ R\}} R< SRel, TRel> Q ≡ (P, Q) ∈ indRelRST SRel TRel

inductive-set (in encoding) indRelRSTPO
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set
  for SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  where
  encR: (SourceTerm S, TargetTerm (\{S\})) ∈ indRelRSTPO SRel TRel | source: (S1, S2) ∈ SRel ⇒ (SourceTerm S1, SourceTerm S2) ∈ indRelRSTPO SRel TRel | target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelRSTPO SRel TRel | trans: [(P, Q) ∈ indRelRSTPO SRel TRel; (Q, R) ∈ indRelRSTPO SRel TRel] ⇒ (P, R) ∈ indRelRSTPO SRel TRel

abbreviation (in encoding) indRelRSTPo
  :: ('procS, 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
  where
  P ∊ {\{R ≲ R\}} R< SRel, TRel> Q ≡ (P, Q) ∈ indRelRSTPO SRel TRel

lemma (in encoding) indRelRSTPO-refl:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes reflS: refl SRel
and reflT: refl TRel
shows refl (indRelRSTPO SRel TRel)
(proof)

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

lemma (in encoding) refl-trans-closure-of-indRelRST:
  fixes SRel :: ('procS x 'procS) set
and TRel :: ('procT x 'procT) set
assumes reflS: refl SRel
and reflT: refl TRel
shows indRelRSTPO SRel TRel = (indRelRST SRel TRel)·
(proof)

inductive-set (in encoding) indRelLST:
  :: ('procS x 'procS) set ⇒ ('procT x 'procT) set
for SRel :: ('procS x 'procS) set
and TRel :: ('procT x '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) indRelLSTPOinfix
  :: ('procS, 'procT) Proc ⇒ ('procS x 'procS) set ⇒ ('procT x 'procT) set
       ⇒ ('procS, 'procT) Proc ⇒ bool (- R LI L <,-> [75, 75, 75, 75] 80)
where
  P R LI L <SRel, TRel> Q ≡ (P, Q) ∈ indRelLST SRel TRel

inductive-set (in encoding) indRelLSTPO:
  :: ('procS x 'procS) set ⇒ ('procT x 'procT) set
for SRel :: ('procS x 'procS) set
and TRel :: ('procT x '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 x 'procS) set ⇒ ('procT x 'procT) set
       ⇒ ('procS, 'procT) Proc ⇒ bool (- ≤ LI L <,-> [75, 75, 75, 75] 80)
where
  P ≤ LI L SRel, TRel> Q ≡ (P, Q) ∈ indRelLSTPO SRel TRel

lemma (in encoding) indRelLSTPO-refl:
  fixes SRel :: ('procS x 'procS) set
and TRel :: ('procT x 'procT) set
assumes reflS: refl SRel
and reflT: refl TRel
shows refl (indRelLSTPO SRel TRel)
(proof)
lemma (in encoding) indRelLSTPO-trans:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  shows trans (indRelLSTPO SRel TRel)
  (proof)

lemma (in encoding) refl-trans-closure-of-indRelLST:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes reflS: refl SRel
  and reflT: refl TRel
  shows indRelLSTPO SRel TRel = (indRelLST SRel TRel)*
  (proof)

inductive-set (in encoding) indRelLST
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set
  for SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  where
  encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelLST SRel TRel |
  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 × 'procS) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
  where
  P ~[.]<SRel,TRel> Q ≡ (P, Q) ∈ indRelLST SRel TRel

lemma (in encoding) indRelLST-symm:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes symmS: symm SRel
  and symmT: symm TRel
  shows symm (indRelLST SRel TRel)
  (proof)

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

abbreviation (in encoding) indRelSTEQinfix
  :: ('procS, 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
  where
  P ~[.]<SRel,TRel> Q ≡ (P, Q) ∈ indRelSTEQ SRel TRel

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

50
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⟩

51
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))²
  ⟨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<SP,TQ>]] 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<TP,TQ>]] 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<SP,TQ>]] 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<TP,TQ>]] 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[[<SP,TQ>]] 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[[<TP,TQ>]] 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:**

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

  - **Proof:**

  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:**

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

  - **Proof:**

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

  - **Proof:**

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

  - **Proof:**

  53
lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-weakly-preserves-barbs:
    fixes SRel :: ('procS × 'procS) set
    and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
    assumes preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)
    and sourceInRel: ∀ S1 S2. (S1, S2) ∈ SRel −→ (SourceTerm S1, SourceTerm S2) ∈ Rel
    shows rel-weakly-preserves-barbs SRel SWB
    (proof)

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

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

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

If indRelRSTPO, indRelLSTPO, or indRelSTPO reflects barbs then so do the corresponding source
and target term relations.

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-reflects-barbs:
    fixes SRel :: ('procS × 'procS) set
    and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
    assumes reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)
    and sourceInRel: ∀ S1 S2. (S1, S2) ∈ SRel −→ (SourceTerm S1, SourceTerm S2) ∈ Rel
    shows rel-reflects-barbs SRel SWB
    (proof)

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

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

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-reflect-barbs:
    fixes SRel :: ('procS × 'procS) set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes reflection: rel-reflects-barbs (indRelSTEQ $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-reflects-barbs $SRel\ SWB$
and rel-reflects-barbs $TRel\ TWB$
(proof)

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-weakly-reflects-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $Rel :: ((\textproc{S}, \textproc{T}) \textproc{Proc} \times ((\textproc{S}, \textproc{T}) \textproc{Proc})$ set
assumes reflection: rel-weakly-reflects-barbs $Rel\ (STCalWB\ SWB\ TWB)$
and sourceInRel: $\forall S1\ S2.\ (S1,\ S2) \in SRel \rightarrow (SourceTerm\ S1,\ SourceTerm\ S2) \in Rel$
shows rel-weakly-reflects-barbs $SRel\ SWB$
(proof)

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes reflection: rel-weakly-reflects-barbs (indRelRSTPO $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-weakly-reflects-barbs $SRel\ SWB$
and rel-weakly-reflects-barbs $TRel\ TWB$
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes reflection: rel-weakly-reflects-barbs (indRelLSTPO $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-weakly-reflects-barbs $SRel\ SWB$
and rel-weakly-reflects-barbs $TRel\ TWB$
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes reflection: rel-weakly-reflects-barbs (indRelSTEQ $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-weakly-reflects-barbs $SRel\ SWB$
and rel-weakly-reflects-barbs $TRel\ TWB$
(proof)

If indRelRSTPO, indRelLSTPO, or indRelSTPO respects barbs then so do the corresponding source
term and target term relations.

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-respect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes respection: rel-respects-barbs (indRelRSTPO $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-respects-barbs $SRel\ SWB$
and rel-respects-barbs $TRel\ TWB$
(proof)

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-respect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes respection: rel-respects-barbs (indRelLSTPO $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-respects-barbs $SRel\ SWB$
and rel-respects-barbs $TRel\ TWB$
(proof)

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-respect-barbs:
fixes $SRel :: (\textproc{S} \times \textproc{S})$ set
and $TRel :: (\textproc{T} \times \textproc{T})$ set
assumes respection: rel-respects-barbs (indRelSTEQ $SRel\ TRel$) ($STCalWB\ SWB\ TWB$)
shows rel-respects-barbs $S_{Rel}$ $SWB$
and rel-respects-barbs $T_{Rel}$ $TWB$

(\proof)

\textbf{lemma (in encoding-wrt-barbs) indRelRSTPO-impl-$S_{Rel}$-and-$T_{Rel}$-weakly-respect-barbs:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes } & \text{respection: rel-weakly-respects-barbs (indRelRSTPO $S_{Rel}$ $T_{Rel}$) (STCalWB $SWB$ $TWB$)} \\
\text{shows } & \text{rel-weakly-respects-barbs $S_{Rel}$ $SWB$ and rel-weakly-respects-barbs $T_{Rel}$ $TWB$} \\
(\proof)
\end{align*}

\textbf{lemma (in encoding-wrt-barbs) indRelLSTPO-impl-$S_{Rel}$-and-$T_{Rel}$-weakly-respect-barbs:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes } & \text{respection: rel-weakly-respects-barbs (indRelLSTPO $S_{Rel}$ $T_{Rel}$) (STCalWB $SWB$ $TWB$)} \\
\text{shows } & \text{rel-weakly-respects-barbs $S_{Rel}$ $SWB$ and rel-weakly-respects-barbs $T_{Rel}$ $TWB$} \\
(\proof)
\end{align*}

\textbf{lemma (in encoding-wrt-barbs) indRelSTEQ-impl-$S_{Rel}$-and-$T_{Rel}$-weakly-respect-barbs:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes } & \text{respection: rel-weakly-respects-barbs (indRelSTEQ $S_{Rel}$ $T_{Rel}$) (STCalWB $SWB$ $TWB$)} \\
\text{shows } & \text{rel-weakly-respects-barbs $S_{Rel}$ $SWB$ and rel-weakly-respects-barbs $T_{Rel}$ $TWB$} \\
(\proof)
\end{align*}

If $T_{Rel}$ is reflexive then $\text{indRelRTPO}$ is a subrelation of $\text{indRelTEQ}$. If $S_{Rel}$ is reflexive then $\text{indRelRTPO}$ is a subrelation of $\text{indRelRTPO}$. Moreover, $\text{indRelRSTPO}$ is a subrelation of $\text{indRelSTEQ}$.

\textbf{lemma (in encoding) indRelRTPO-to-indRelTEQ:}
\begin{align*}
\text{fixes } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{and } & P \ Q :: (\text{procS}, \text{procT}) \text{ Proc} \\
\text{assumes } & \text{rel: } P \leq [\ ]RT < T_{Rel} > Q \\
\text{and } & \text{reflT: refl } T_{Rel} \\
\text{shows } & P \sim [\ ]T < T_{Rel} > Q \\
(\proof)
\end{align*}

\textbf{lemma (in encoding) indRelRTPO-to-indRelRSTPO:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{and } & P \ Q :: (\text{procS}, \text{procT}) \text{ Proc} \\
\text{assumes } & \text{rel: } P \leq [\ ]RT < T_{Rel} > Q \\
\text{and } & \text{reflS: refl } S_{Rel} \\
\text{shows } & P \leq [\ ]R < S_{Rel}, T_{Rel} > Q \\
(\proof)
\end{align*}

\textbf{lemma (in encoding) indRelRSTPO-to-indRelSTEQ:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{and } & P \ Q :: (\text{procS}, \text{procT}) \text{ Proc} \\
\text{assumes } & \text{rel: } P \leq [\ ]R < S_{Rel}, T_{Rel} > Q \\
\text{shows } & P \sim [\ ]< S_{Rel}, T_{Rel} > Q \\
(\proof)
\end{align*}

If $\text{indRelRTPO}$ is a bisimulation and $S_{Rel}$ is a reflexive bisimulation then also $\text{indRelRSTPO}$ is a bisimulation.

\textbf{lemma (in encoding) indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation:}
\begin{align*}
\text{fixes } & S_{Rel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & T_{Rel} :: (\text{procT} \times \text{procT}) \text{ set}
\end{align*}
assumes \( \text{bisim}_T: \text{weak-reduction-bisimulation} (\text{indRelRTPO} \ T\text{Rel}) (\text{STCal} \ Source \ Target) \)
and \( \text{bisim}_S: \text{weak-reduction-bisimulation} \ S\text{Rel} \ Source \)
and \( \text{refl}_S: \text{refl} \ S\text{Rel} \)
shows \( \text{weak-reduction-bisimulation} (\text{indRelRSTPO} \ S\text{Rel} \ T\text{Rel}) (\text{STCal} \ Source \ Target) \)

\[
\text{(proof)}
\]

\text{end}

theory \text{SuccessSensitiveness}

\text{imports} \text{SourceTargetRelation}

\text{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 \( a \).

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-preserves-barb-set :: } \text{`barbs set } \Rightarrow \text{ bool where }
\text{enc-weakly-preserves-barb-set Barbs } \equiv \text{enc-preserves-binary-pred} (\lambda P \ a. \ a \in \text{Barbs} \land P \downarrow. a)
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-preserves-barbs :: bool where }
\text{enc-weakly-preserves-barbs } \equiv \text{enc-preserves-binary-pred} (\lambda P \ a. \ P \downarrow. a)
\]

\[
\text{lemma (in encoding-wrt-barbs) enc-weakly-preserves-barbs-and-barb-set: }
\text{shows enc-weakly-preserves-barbs } = (\forall \text{Barbs}. \text{enc-weakly-preserves-barb-set Barbs})
\text{(proof)}
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-reflects-barb-set :: } \text{`barbs set } \Rightarrow \text{ bool where }
\text{enc-weakly-reflects-barb-set Barbs } \equiv \text{enc-reflects-binary-pred} (\lambda P \ a. \ a \in \text{Barbs} \land P \downarrow. a)
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-reflects-barbs :: bool where }
\text{enc-weakly-reflects-barbs } \equiv \text{enc-reflects-binary-pred} (\lambda P \ a. \ P \downarrow. a)
\]

\[
\text{lemma (in encoding-wrt-barbs) enc-weakly-reflects-barbs-and-barb-set: }
\text{shows enc-weakly-reflects-barbs } = (\forall \text{Barbs}. \text{enc-weakly-reflects-barb-set Barbs})
\text{(proof)}
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-respects-barb-set :: } \text{`barbs set } \Rightarrow \text{ bool where }
\text{enc-weakly-respects-barb-set Barbs } \equiv \text{enc-weakly-preserves-barb-set Barbs} \land \text{enc-weakly-reflects-barb-set Barbs}
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-weakly-respects-barbs :: bool where }
\text{enc-weakly-respects-barbs } \equiv \text{enc-weakly-preserves-barbs } \land \text{enc-weakly-reflects-barbs}
\]

\[
\text{lemma (in encoding-wrt-barbs) enc-weakly-respects-barbs-and-barb-set: }
\text{shows enc-weakly-respects-barbs } = (\forall \text{Barbs}. \text{enc-weakly-respects-barb-set Barbs})
\text{(proof)}
\]

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 \( a \).

\[
\text{abbreviation (in encoding-wrt-barbs) enc-preserves-barb-set :: } \text{`barbs set } \Rightarrow \text{ bool where }
\text{enc-preserves-barb-set Barbs } \equiv \text{enc-preserves-binary-pred} (\lambda P \ a. \ a \in \text{Barbs} \land P \downarrow. a)
\]

\[
\text{abbreviation (in encoding-wrt-barbs) enc-preserves-barbs :: bool where }
\]
\[ \text{enc-preserves-barbs} \equiv \text{enc-preserves-binary-pred} \ (\lambda P \ a. \ P \downarrow a) \]

**Lemma (in encoding-wrt-barbs)** \(\text{enc-preserves-barbs-and-barb-set}::\)

\[ \text{shows} \ \text{enc-preserves-barbs} = (\forall \text{Barbs. enc-preserves-barb-set Barbs}) \]

**(proof)**

**Abbreviation (in encoding-wrt-barbs)** \(\text{enc-reflects-barb-set} :: \text{barbs set} \Rightarrow \text{bool} \) where

\[ \text{enc-reflects-barb-set} \ \text{Barbs} \equiv \text{enc-reflects-binary-pred} (\lambda P \ a. \ a \in \text{Barbs} \land P \downarrow a) \]

**Lemma (in encoding-wrt-barbs)** \(\text{enc-reflects-barbs-and-barb-set}::\)

\[ \text{shows} \ \text{enc-reflects-barbs} = (\forall \text{Barbs. enc-reflects-barb-set Barbs}) \]

**(proof)**

**Abbreviation (in encoding-wrt-barbs)** \(\text{enc-reflects-barbs} :: \text{bool} \) where

\[ \text{enc-reflects-barbs} \equiv \text{enc-reflects-binary-pred} (\lambda P \ a. \ P \downarrow a) \]

**Lemma (in encoding-wrt-barbs)** \(\text{enc-reflects-barbs-and-barb-set}::\)

\[ \text{shows} \ \text{enc-reflects-barbs} = (\forall \text{Barbs. enc-reflects-barb-set Barbs}) \]

**(proof)**

**Abbreviation (in encoding-wrt-barbs)** \(\text{enc-reflects-barbs} :: \text{bool} \) where

\[ \text{enc-reflects-barbs} \equiv \text{enc-reflects-binary-pred} (\lambda P \ a. \ P \downarrow a) \]

**Lemma (in encoding-wrt-barbs)** \(\text{enc-reflects-barbs-and-barb-set}::\)

\[ \text{shows} \ \text{enc-reflects-barbs} = (\forall \text{Barbs. enc-reflects-barb-set Barbs}) \]

**(proof)**

An encoding (weakly) preserves barbs iff (1) there exists a relation, like \(\text{indRelR}\), that relates source terms and their literal translations and preserves (reachability/existence of barbs, or (2) there exists a relation, like \(\text{indRelL}\), that relates literal translations and their source terms and reflects (reachability/existence of barbs.

**Lemma (in encoding-wrt-barbs)** \(\text{enc-weakly-preserves-barb-set-iff-source-target-rel}::\)

\[ \text{fixes} \ \text{Barbs} :: \text{barbs set} \]

\[ \text{and} \ \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \]

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

**(proof)**

**Lemma (in encoding-wrt-barbs)** \(\text{enc-weakly-preserves-barbs-iff-source-target-rel}::\)

\[ \text{fixes} \ \text{Barbs} :: \text{barbs set} \]

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

**(proof)**

**Lemma (in encoding-wrt-barbs)** \(\text{enc-preserves-barb-set-iff-source-target-rel}::\)

\[ \text{fixes} \ \text{Barbs} :: \text{barbs set} \]

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

**(proof)**

**Lemma (in encoding-wrt-barbs)** \(\text{enc-preserves-barbs-iff-source-target-rel}::\)

\[ \text{fixes} \ \text{Barbs} :: \text{barbs set} \]

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

**(proof)**

An encoding (weakly) reflects barbs iff (1) there exists a relation, like \(\text{indRelR}\), that relates source terms and their literal translations and reflects (reachability/existence of barbs, or (2) there exists a relation, like \(\text{indRelL}\), 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 Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-weakly-reflects-barb-set Rel (STCalWB SWB TWB) Barbs)`
  - `(proof)`

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

- **Shows** `enc-weakly-reflects-barbs`
  - `\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-weakly-reflects-barbs 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 Barbs`
  - `\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-reflects-barb-set Rel (STCalWB SWB TWB) Barbs)`
  - `(proof)`

Accordingly 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 Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) Barbs)`
  - `(proof)`

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

- **Shows** `enc-weakly-respects-barbs`
  - `\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-weakly-respects-barbs 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 Barbs`
  - `\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-respects-barb-set Rel (STCalWB SWB TWB) Barbs)`
  - `(proof)`

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

- **Shows** `enc-respects-barbs`
  - `\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
  \land rel-respects-barbs 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-weakenly-respects-barb-set {success} = (\forall S. S \downarrow_{\ll＜SWB>success} \iff [S] \downarrow_{\ll＜TWB>success})
  ⟨proof⟩

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

lemma (in encoding-wrt-barbs) success-sensitive-iff-source-target-rel-respects-success:
  fixes success :: 'barbs
  shows enc-respects-barb-set {success} = (
    \exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
    \land rel-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 = enc-preserves-pred (\lambda P. P \mapsto_{ST\omega})

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

abbreviation (in encoding) enc-reflects-divergence :: bool where
  enc-reflects-divergence = enc-reflects-pred (\lambda P. P \mapsto_{ST\omega})

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

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

abbreviation rel-reflects-divergence :: ('proc \times 'proc) set \Rightarrow 'proc processCalculus \Rightarrow bool
  where
  rel-reflects-divergence Rel Cal = rel-reflects-pred Rel (\lambda P. P \mapsto_{\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 = enc-respects-pred (\lambda P. P \mapsto_{ST\omega})
lemma (in encoding) divergence-respection-cond:
  shows enc-respects-divergence = (∀ S. [S] ⟷ (Target)ω ↔ S ⟷ (Source)ω)
  ⟨proof⟩

abbreviation rel-respects-divergence
  :: (′proc × ′proc) set ⇒ ′proc processCalculus ⇒ bool
  where
  rel-respects-divergence Rel Cal = rel-respects-pred Rel (λP. P ⟷ (Cal)ω)

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))
  ⟨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
operational-complete TRel ≡ ∀ S S’. S ḅ→ Source S’ ḅ→ (∃ T. [S] ḅ→ Target T ∧ ([S’], T) ∈ 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
strongly-operational-complete TRel ≡ ∀ S S’. S ḅ→ Source S’ ḅ→ (∃ T. [S] ḅ→ Target T ∧ ([S’], T) ∈ 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
weakly-operational-sound TRel ≡ ∀ S T. [S] ḅ→ Target T ḅ→ (∃ S’ T’. S ḅ→ Source S’ ∧ T ḅ→ Target T’ ∧ ([S’], T’) ∈ 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
operational-sound TRel ≡ ∀ S T. [S] ḅ→ Target T ḅ→ (∃ S’ T’. S ḅ→ Source S’ ∧ ([S’], T) ∈ 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
strongly-operational-sound TRel ≡ ∀ S T. [S] ḅ→ Target T ḅ→ (∃ S’ T’. S ḅ→ Source S’ ∧ ([S’], T) ∈ TRel)

An encoding is weakly operational corresponding modulo TRel if it is operational complete and weakly operational sound modulo TRel.

**abbreviation** (in encoding) weakly-operational-corresponding :: (’procT × ’procT) set ⇒ bool where
weakly-operational-corresponding TRel ≡ operational-complete TRel ∧ 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
operational-corresponding TRel ≡ operational-complete TRel ∧ operational-sound TRel

An encoding is strongly operational corresponding modulo TRel if it is strongly operational complete and strongly operational sound modulo TRel.

**abbreviation** (in encoding) strongly-operational-corresponding :: (’procT × ’procT) set ⇒ bool where
strongly-operational-corresponding TRel ≡ strongly-operational-complete TRel ∧ strongly-operational-sound TRel
8.1 Trivial Operational Correspondence Results

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

**Lemma** (in encoding) operational-correspondence-modulo-all-relation:
- shows operational-complete \{\langle T_1, T_2 \rangle, \text{True}\}
- and weakly-operational-sound \{\langle T_1, T_2 \rangle, \text{True}\}
- and operational-sound \{\langle T_1, T_2 \rangle, \text{True}\}

**(proof)**

**Lemma** all-relation-is-weak-reduction-bisimulation:
- fixes Cal :: 'a processCalculus
- shows weak-reduction-bisimulation \{\langle a, b \rangle, \text{True}\} Cal

**(proof)**

**Lemma** (in encoding) operational-correspondence-modulo-some-target-relation:
- shows \exists TRel. weakly-operational-corresponding TRel
- and \exists TRel. operational-corresponding TRel
- and \exists TRel. weakly-operational-corresponding TRel \land weak-reduction-bisimulation TRel Target
- and \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-correspondence-modulo-some-target-relation:
- shows \exists TRel. strongly-operational-corresponding TRel
- and \exists TRel. strongly-operational-corresponding TRel
- and \exists TRel. weakly-operational-corresponding TRel \land weak-reduction-bisimulation TRel Target
- and \exists TRel. operational-corresponding 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 :: \{\langle 'procS', 'procT' \rangle Proc \times \langle 'procS', 'procT' \rangle Proc\} set
- and TRel :: \langle 'procT' \times 'procT' \rangle set
- assumes A1: \forall S. (SourceTerm S, TargetTerm \langle [S] \rangle) \in Rel
- and A2: \forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^*
- and A3: weak-reduction-simulation Rel (STCal Source Target)
- shows operational-complete (TRel^*)

**(proof)**

**Lemma** (in encoding) OCom-iff-indRelRTPO-is-weak-reduction-simulation:
- fixes TRel :: \langle 'procT' \times 'procT' \rangle set
- shows (operational-complete (TRel^*) \land weak-reduction-simulation (TRel^*) Target)
  = weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)

**(proof)**

**Lemma** (in encoding) OCom-iff-weak-reduction-simulation:
- fixes TRel :: \langle 'procT' \times 'procT' \rangle set
- shows (operational-complete (TRel^*) \land weak-reduction-simulation (TRel^*) Target)
  = (\exists Rel. (\forall S. (SourceTerm S, TargetTerm \langle [S] \rangle) \in Rel)
  \land (\forall T_1 T_2. (T_1, T_2) \in TRel \rightarrow (TargetTerm T_1, TargetTerm T_2) \in Rel)
  \land (\forall T_1 T_2. (TargetTerm T_1, TargetTerm T_2) \in Rel \rightarrow (T_1, T_2) \in TRel^*)
\[ (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*) \]
\[ \wedge \text{weak-reduction-simulation } \text{Rel} \quad \text{(STCal Source Target)} \]

\textbf{(proof)}

An encoding is strong operational complete modulo a strong simulation on target terms TRel iff there is a relation, like \text{indRelRTPO}, that relates at least all source terms to their literal translations, includes TRel, and is a strong simulation.

\textbf{lemma (in encoding) strong-reduction-simulation-impl-SOCom:}
\begin{itemize}
  \item \text{fixes Rel} \mathrel{::} (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{set}
  \item TRel \mathrel{::} ('procT \times 'procT) \text{set}
  \item \text{assumes A1: } \forall S. \text{(SourceTerm } S, \text{TargetTerm } ([S]) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*)
  \item \text{shows strongly-operational-complete } (\text{TRel}^*)
\end{itemize}
\textbf{(proof)}

\textbf{lemma (in encoding) SOCom-iff-indRelRTPO-is-strong-reduction-simulation:}
\begin{itemize}
  \item \text{fixes TRel} \mathrel{::} ('\text{procT} \times '\text{procT}) \text{set}
  \item \text{shows (strongly-operational-complete } (\text{TRel}^*)
  \item = \text{strong-reduction-simulation } (\text{indRelRTPO } \text{TRel}) \quad \text{(STCal Source Target)}
\end{itemize}
\textbf{(proof)}

\textbf{lemma (in encoding) target-relation-from-source-target-relation:}
\begin{itemize}
  \item \text{fixes str} \mathrel{::} (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{set}
  \item \text{assumes } \exists TRel. \forall S T. \text{(SourceTerm } S, \text{TargetTerm } ([S]) \in \text{Rel} \rightarrow \text{TargetTerm } T \in \text{Rel}^*)
  \item \text{shows } \exists TRel. \forall S T. \text{(SourceTerm } S, \text{TargetTerm } ([S]) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*)
\end{itemize}
\textbf{(proof)}

\textbf{lemma (in encoding) SOCom-modulo-TRel-iff-strong-reduction-simulation:}
\begin{itemize}
  \item \text{shows (3 TRel. strongly-operational-complete } (\text{TRel}^*)
  \item = (\exists TRel. \forall S T. \text{(SourceTerm } S, \text{TargetTerm } ([S]) \in \text{Rel} \rightarrow \text{TargetTerm } T \in \text{Rel}^*)
  \item \wedge \text{strong-reduction-simulation } \text{Rel} \quad \text{(STCal Source Target)}
\end{itemize}
\textbf{(proof)}

\textbf{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.

\textbf{lemma (in encoding) weak-reduction-contrasimulation-impl-WOSou:}
\begin{itemize}
  \item \text{fixes Rel} \mathrel{::} (('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}) \text{set}
  \item TRel \mathrel{::} ('procT \times 'procT) \text{set}
  \item \text{assumes A1: } \forall S. \text{(SourceTerm } S, \text{TargetTerm } ([S]) \in \text{Rel}
\end{itemize}
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** A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel

- **and** A2: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*

**shows** weakly-operational-sound (TRel*)

**(proof)**

**Lemma (in encoding) OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation:**

- **fixes** TRel :: (′procT × ′procT) set

**shows** (operational-sound (TRel*))

= weak-reduction-simulation ((TRel+)−1 Target)

= weak-reduction-simulation ((indRelRTPO TRel)−1) (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** A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel

- **and** A2: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*

**shows** strongly-operational-sound (TRel*)

**(proof)**

**Lemma (in encoding) SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation:**

- **fixes** TRel :: (′procT × ′procT) set

**shows** (strongly-operational-sound (TRel*))

= strong-reduction-simulation ((TRel+)−1 Target)

= strong-reduction-simulation ((indRelRTPO TRel)−1) (STCal Source Target)

**(proof)**

**Lemma (in encoding) SOSou-iff-strong-reduction-simulation:**

- **fixes** TRel :: (′procT × ′procT) set

**shows** (strongly-operational-sound (TRel*) ∧ strong-reduction-simulation ((TRel+)−1 Target)
\[(\exists S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})\]
\[\land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel})\]
\[\land (\forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+)\]
\[\land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+)\]
\[\land \text{strong-reduction-simulation} (\text{Rel}^{-1}) (\text{STCal} \text{Source Target})\]

**proof**

**lemma** (in **encoding**) **SOSou-modulo-TRel-iff-strong-reduction-simulation**:

**shows** (\(3 \text{TRel. strongly-operational-sound (TRel^*)}\)
\[\land \text{strong-reduction-simulation} (\text{TRel}^+)^{-1} \text{Target}\]
\[= (\exists S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})\]
\[\land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}^-)\]
\[\land \text{strong-reduction-simulation} (\text{Rel}^{-1}) (\text{STCal} \text{Source Target})\]

**proof**

**8.5 Weak Operational Correspondence vs Correspondence Similarity**

If there exists a relation that relates at least all source terms and their literal translations, includes TRel, and is a correspondence simulation then the encoding is weakly operational corresponding w.r.t. TRel.

**lemma** (in **encoding**) **weak-reduction-correspondence-simulation-impl-WOC**:

**fixes** \(\text{Rel} :: ((\text{procS}, \text{procT}) \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc}) \text{set}\)
\[\text{and} \ TRel :: (\text{procT} \times \text{procT}) \text{set}\]
**assumes** \(\text{enc} :: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}\)
\[\text{and} \ TRel :: (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}^-)\]
\[\text{and} \ cs :: \text{weak-reduction-correspondence-simulation} \text{Rel} (\text{STCal} \text{Source Target})\]

**shows** \(\text{weakly-operational-corresponding} (\text{TRel}^*)\)

**proof**

An encoding is weakly operational corresponding w.r.t. a correspondence simulation on target terms TRel iff there exists a relation, like indRelRTPO, that relates at least all source terms and their literal translations, includes TRel, and is a correspondence simulation.

**lemma** (in **encoding**) **WOC-iff-indRelRTPO-is-reduction-correspondence-simulation**:

**fixes** \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{set}\)
**shows** \(\text{weakly-operational-corresponding} (\text{TRel}^*)\)
\[= \text{weak-reduction-correspondence-simulation} (\text{indRelRTPO TRel}) (\text{STCal} \text{Source Target})\]

**proof**

**lemma** (in **encoding**) **WOC-iff-reduction-correspondence-simulation**:

**fixes** \(\text{TRel} :: (\text{procT} \times \text{procT}) \text{set}\)
**shows** \(\text{weakly-operational-corresponding} (\text{TRel}^*)\)
\[\land \text{weak-reduction-correspondence-simulation} (\text{TRel}^+) \text{Target}\]
\[= (\exists S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})\]
\[\land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel})\]
\[\land (\forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+)\]
\[\land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+)\]
\[\land \text{weak-reduction-correspondence-simulation} \text{Rel} (\text{STCal} \text{Source Target})\]

**proof**

**lemma** **rel-includes-TRel-module-preorder**:

**fixes** \(\text{Rel} :: ((\text{procS}, \text{procT}) \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc}) \text{set}\)
\[\text{and} \ TRel :: (\text{procT} \times \text{procT}) \text{set}\]
**assumes** \(\text{transT} :: \text{trans TRel}\)
\[\text{shows} ((\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel})\]
\[\land (\forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+)\]
\[= (\text{TRel} = \{(T_1, T_2). (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel}\})\]

**proof**
\textbf{lemma (in encoding) }\text{WOC-wrt-preorder-iff-reduction-correspondence-simulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{shows} \ (\text{weakly-operational-corresponding} \ TRel \wedge \text{preorder} \ TRel
\wedge \text{weak-reduction-correspondence-simulation} \ TRel \text{ Target})
= (\exists \text{ Rel}. \ (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} ([S])) \in \text{Rel})
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{Rel}\}
\wedge (\forall S T. (\text{SourceTerm} \ S, \text{TargetTerm} \ T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})
\wedge \text{preorder Rel}
\wedge \text{weak-reduction-correspondence-simulation} \ Rel \ (\text{STCal Source Target}))
\end{quote}

\section{8.6 \text{(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 \text{indRelRTPO}, that relates at least all source terms and their literal translations, includes TRel, and is a weak bisimulation. Thus this variant of operational correspondence ensures that source terms and their translations are weak bisimilar.

\textbf{lemma (in encoding) }\text{OC-iff-indRelRTPO-is-weak-reduction-bisimulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{shows} \ (\text{operational-corresponding} \ (TRel^*) \wedge \text{weak-reduction-bisimulation} \ (TRel^+) \text{ Target})
= \text{weak-reduction-bisimulation} \ (\text{indRelRTPO TRel}) \ (\text{STCal Source Target})
\end{quote}

\textbf{lemma (in encoding) }\text{OC-iff-weak-reduction-bisimulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{shows} \ (\text{operational-corresponding} \ (TRel^*) \wedge \text{weak-reduction-bisimulation} \ (TRel^+) \text{ Target})
= (\exists \text{ Rel}. \ (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} ([S])) \in \text{Rel})
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{Rel}\}
\wedge (\forall S T. (\text{SourceTerm} \ S, \text{TargetTerm} \ T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+)
\wedge \text{preorder Rel}
\wedge \text{weak-reduction-bisimulation} \ Rel \ (\text{STCal Source Target}))
\end{quote}

\textbf{lemma (in encoding) }\text{OC-wrt-preorder-iff-weak-reduction-bisimulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{shows} \ (\text{operational-corresponding} \ TRel \wedge \text{preorder} \ TRel
\wedge \text{weak-reduction-bisimulation} \ TRel \text{ Target})
= (\exists \text{ Rel}. \ (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} ([S])) \in \text{Rel})
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{Rel}\}
\wedge (\forall S T. (\text{SourceTerm} \ S, \text{TargetTerm} \ T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})
\wedge \text{preorder Rel}
\wedge \text{weak-reduction-bisimulation} \ Rel \ (\text{STCal Source Target}))
\end{quote}

\textbf{lemma (in encoding) }\text{OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{assumes} \ \text{eqT: equivalence} \ TRel
\text{shows} \ (\text{operational-corresponding} \ TRel \wedge \text{weak-reduction-bisimulation} \ TRel \text{ Target}) \leftrightarrow
\text{weak-reduction-bisimulation} \ (\text{indRelTEQ TRel}) \ (\text{STCal Source Target})
\end{quote}

\textbf{lemma (in encoding) }\text{OC-wrt-equivalence-iff-weak-reduction-bisimulation:}
\begin{quote}
\text{fixes} \ TRel :: \ ('\text{proc}T \times '\text{proc}T) \text{ set}
\text{assumes} \ \text{eqT: equivalence} \ TRel
\text{shows} \ (\text{operational-corresponding} \ TRel \wedge \text{weak-reduction-bisimulation} \ TRel \text{ Target}) \leftrightarrow (\exists \text{ Rel}. \ (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} ([S])) \in \text{Rel} \wedge (\text{TargetTerm} ([S]), \text{SourceTerm} \ S) \in \text{Rel})
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{Rel}\}
\wedge \text{trans Rel} \wedge \text{weak-reduction-bisimulation} \ Rel \ (\text{STCal Source Target}))
\end{quote}
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:**

- **F**ixes TRel :: ('procT x 'procT) set
- **S**hows (strongly-operational-corresponding (TRel*)
  ∧ strong-reduction-bisimulation (TRel*) Target)
  = strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

**Proof**

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

- **F**ixes TRel :: ('procT x 'procT) set
- **S**hows (strongly-operational-corresponding (TRel*)
  ∧ strong-reduction-bisimulation (TRel*) Target)
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈ TRel → [(TargetTerm T1, TargetTerm T2)] ∈ Rel)
  ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel+)
  ∧ (∃ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*)
  ∧ strongly-operational-corresponding Rel (STCal Source Target))

**Proof**

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

- **F**ixes TRel :: ('procT x 'procT) set
- **S**hows (strongly-operational-corresponding TRel ∧ preorder TRel
  ∧ strong-reduction-bisimulation TRel Target)
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∃ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ preorder Rel
  ∧ strongly-reduction-bisimulation Rel (STCal Source Target))

**Proof**

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

- **S**hows (∃ TRel. strongly-operational-corresponding (TRel*)
  ∧ strong-reduction-bisimulation (TRel*) Target)
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ strongly-operational-corresponding Rel (STCal Source Target))

**Proof**

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

- **F**ixes TRel :: ('procT x 'procT) set
- **A**ssumes eqT: equivalence TRel
- **S**hows (strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target)
  ↔ strongly-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)

**Proof**

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

- **F**ixes TRel :: ('procT x 'procT) set
- **A**ssumes eqT: equivalence TRel
- **S**hows (strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target)
  ↔ (∀ Rel.
  (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
  ∧ TRel = {([T1, T2], (TargetTerm T1, TargetTerm T2) ∈ Rel}
  ∧ trans Rel ∧ strongly-reduction-bisimulation Rel (STCal Source Target))

**Proof**
9 Full Abstraction

An encoding is fully abstract w.r.t. some source term relation $S_{Rel}$ and some target term relation $T_{Rel}$ if two source terms $S_1$ and $S_2$ form a pair $(S_1, S_2)$ in $S_{Rel}$ iff their literal translations form a pair $(\text{enc } S_1, \text{enc } S_2)$ in $T_{Rel}$.

**Abbreviation (in encoding)**

\[ \text{fully-abstract} :: (procS \times procS) \Rightarrow (procT \times procT) \Rightarrow \text{bool} \]

\[ \text{fully-abstract } S_{Rel} \ T_{Rel} \equiv \forall S_1 S_2. (S_1, S_2) \in S_{Rel} \iff ([S_1], [S_2]) \in T_{Rel} \]

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)**

\[ \text{inj-enc-is-fully-abstract-wrt-identities} : \]

\[ \text{assumes injectivity: } \forall S_1 S_2. [S_1] = [S_2] \rightarrow S_1 = S_2 \]

\[ \text{shows fully-abstract } \{ (S_1, S_2). S_1 = S_2 \} \{ (T_1, T_2). T_1 = T_2 \} \]

Each encoding is fully abstract w.r.t. the empty relation on the source and the target.

**Lemma (in encoding)**

\[ \text{fully-abstract-wrt-empty-relation} : \]

\[ \text{shows fully-abstract } \{ \} \{ \} \]

Similarly, each encoding is fully abstract w.r.t. the all-relation on the source and the target.

**Lemma (in encoding)**

\[ \text{fully-abstract-wrt-all-relation} : \]

\[ \text{shows fully-abstract } \{ (S_1, S_2). \text{True} \} \{ (T_1, T_2). \text{True} \} \]

If the encoding is injective then for each source term relation $R_{S}$ there exists a target term relation $R_{T}$ such that the encoding is fully abstract w.r.t. $R_{S}$ and $R_{T}$.

**Lemma (in encoding)**

\[ \text{fully-abstract-wrt-source-relation} : \]

\[ \text{fixes } R_{S} :: (procS \times procS) \text{ set} \]

\[ \text{assumes injectivity: } \forall S_1 S_2. [S_1] = [S_2] \rightarrow S_1 = S_2 \]

\[ \text{shows } \exists R_{T}. \text{fully-abstract } R_{S} R_{T} \]

If all source terms that are translated to the same target term are related by a trans source term relation $R_{S}$, then there exists a target term relation $R_{T}$ such that the encoding is fully abstract w.r.t. $R_{S}$ and $R_{T}$.

**Lemma (in encoding)**

\[ \text{fully-abstract-wrt-trans-source-relation} : \]

\[ \text{fixes } R_{S} :: (procS \times procS) \text{ set} \]

\[ \text{assumes } \text{encRelS: } \forall S_1 S_2. [S_1] = [S_2] \rightarrow (S_1, S_2) \in R_{S} \]

\[ \text{and } \text{transS: } \text{trans } R_{S} \]

\[ \text{shows } \exists R_{T}. \text{fully-abstract } R_{S} R_{T} \]

**Lemma (in encoding)**

\[ \text{fully-abstract-wrt-trans-closure-of-source-relation} : \]

\[ \text{fixes } R_{S} :: (procS \times procS) \text{ set} \]

\[ \text{assumes } \text{encRelS: } \forall S_1 S_2. [S_1] = [S_2] \rightarrow (S_1, S_2) \in R_{S}^{+} \]

\[ \text{shows } \exists R_{T}. \text{fully-abstract } (R_{S}^{+}) R_{T} \]
For every encoding and every target term relation RelT there exists a source term relation RelS such that the encoding is fully abstract w.r.t. RelS and RelT.

**Lemma (in encoding) fully-abstract-wrt-target-relation:**
- **Fixes** RelT :: (procT x procT) set
- **Shows** ∃ RelS. fully-abstract RelS RelT

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

**Lemma (in encoding) full-abstraction-and-trans-relation-contains-SRel-impl-TRel:**
- **Fixes** Rel :: (procS x procT) Proc x (procS x procT) Proc set
- **And** SRel :: (procS x procS) set
- **And** TRel :: (procT x procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** srel: SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
- **And** trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
- **Shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

**(proof)**

**Lemma (in encoding) full-abstraction-and-trans-relation-contains-TRel-impl-SRel:**
- **Fixes** Rel :: (procS x procT) Proc x (procS x procT) Proc set
- **And** SRel :: (procS x procS) set
- **And** TRel :: (procT x procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** trel: ∀ S1 S2. ([S1], [S2]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
- **And** trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
- **Shows** SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}

**(proof)**

**Lemma (in encoding) full-abstraction-impl-trans-relation-contains-SRel-iff-TRel:**
- **Fixes** Rel :: (procS x procT) Proc x (procS x procT) Proc set
- **And** SRel :: (procS x procS) set
- **And** TRel :: (procT x procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
- **Shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

**(proof)**

**Lemma (in encoding) full-abstraction-impl-trans-relation-contains-SRel-iff-TRel-encRL:**
- **Fixes** Rel :: (procS x procT) Proc x (procS x procT) Proc set
- **And** SRel :: (procS x procS) set
- **And** TRel :: (procT x procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** encL: ∀ S. (TargetTerm ([S]), SourceTerm S) ∈ Rel
- **And** trans: trans Rel
- **Shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

**(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 \in \SP P \land \SQ \in \SQ Q \land (SP, SQ) \in SRel\} Pred
≡ rel-preserves-binary-pred \{(P, Q). \exists \SP SQ. [[SP]] \in T P \land [[SQ]] \in T Q \land ([SP], [SQ]) \in TRel\} Pred

\langle proof \rangle

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-pred Pred
shows rel-reflects-binary-pred \{(P, Q). \exists \SP SQ. \SP \in \SP P \land \SQ \in \SQ Q \land (SP, SQ) \in SRel\} Pred
≡ rel-reflects-binary-pred \{(P, Q). \exists \SP SQ. [[SP]] \in T P \land [[SQ]] \in T Q \land ([SP], [SQ]) \in TRel\} Pred

\langle proof \rangle

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflected:

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-reflected-binary-pred \{(P, Q). \exists \SP SQ. \SP \in \SP P \land \SQ \in \SQ Q \land (SP, SQ) \in SRel\} Pred
≡ rel-reflected-binary-pred \{(P, Q). \exists \SP SQ. [[SP]] \in T P \land [[SQ]] \in T Q \land ([SP], [SQ]) \in TRel\} Pred

\langle proof \rangle

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflected:

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-reflected-binary-pred \{(P, Q). \exists \SP SQ. \SP \in \SP P \land \SQ \in \SQ Q \land (SP, SQ) \in SRel\} Pred
≡ rel-reflected-binary-pred \{(P, Q). \exists \SP SQ. [[SP]] \in T P \land [[SQ]] \in T Q \land ([SP], [SQ]) \in TRel\} Pred

\langle proof \rangle

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflected:

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-reflected-binary-pred \{(P, Q). \exists \SP SQ. \SP \in \SP P \land \SQ \in \SQ Q \land (SP, SQ) \in SRel\} Pred
≡ rel-reflected-binary-pred \{(P, Q). \exists \SP SQ. [[SP]] \in T P \land [[SQ]] \in T Q \land ([SP], [SQ]) \in TRel\} Pred

\langle proof \rangle

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:
\textbf{Fixes} $\text{Rel} :: (\langle \text{procS}, \text{procT} \rangle \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc})$ set
\textbf{Assumes} $\text{enc}: \forall S. (\text{SourceTerm S}, \text{TargetTerm (S)}) \in \text{Rel}$
\text{and} $\text{trans: trans Rel}$
\textbf{Shows} fully-abstract $(\langle S1, S2 \rangle, \text{SourceTerm S1, SourceTerm S2} \rangle \in \text{Rel})$
\text{(proof)}

\textbf{Lemma (in encoding) source-target-relation-impl-full-abstraction-wrt-trans-closures:}
\textbf{Fixes} $\text{Rel} :: (\langle \text{procS}, \text{procT} \rangle \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc})$ set
\textbf{Assumes} $\text{enc}: \forall S. (\text{SourceTerm S}, \text{TargetTerm (S)}) \in \text{Rel}$
\text{and} $\text{SRel} :: (\text{procS} \times \text{procS})$ set
\text{and} $\text{TRel} :: (\text{procT} \times \text{procT})$ set
\textbf{Shows} fully-abstract $\text{SRel TRel}$
\text{(proof)}

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

\textbf{Lemma (in encoding) full-abstraction-impl-indRelRTPO-to-SRel-and-TRel:}
\textbf{Fixes} $\text{SRel} :: (\langle \text{procS}, \text{procT} \rangle \text{Proc} \times \text{procS})$ set
\text{and} $\text{TRel} :: (\langle \text{procT} \times \text{procT} \rangle \text{Proc}$
\textbf{Assumes} fullAbs: fully-abstract $\text{SRel TRel}$
\text{and} $\text{rel}: P \leq_{\text{RTPO}} Q$
\textbf{Shows} $\forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow ([SP], [SQ]) \in \text{TRel}^+$
\text{and} $\forall SP TQ. \exists S P \land TQ \in T Q \rightarrow ([SP], [TQ]) \in \text{TRel}^+$
\text{(proof)}

\textbf{Lemma (in encoding) full-abstraction-wrt-preorders-impl-indRelSTEO-to-SRel-and-TRel:}
\textbf{Fixes} $\text{SRel} :: (\langle \text{procS} \times \text{procS} \rangle)$ set
\text{and} $\text{TRel} :: (\langle \text{procT} \times \text{procT} \rangle \text{Proc}$
\textbf{Assumes} $\text{fA}: \text{fully-abstract SRel TRel}$
\text{and} $\text{transT: trans TRel}$
\text{and} $\text{reflS: refl SRel}$
\text{and} $\text{rel}: P \sim ([\text{SRel} \text{TRel}]) Q$
\textbf{Shows} $\forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow (SP, SQ) \in \text{SRel}$
\textbf{and} $\forall SP SQ. \exists S P \land SQ \in S Q \rightarrow ([SP], [SQ]) \in \text{TRel}$
\text{and} $\forall SP TQ. \exists S P \land TQ \in T Q \rightarrow ([SP], [TQ]) \in \text{TRel}$
\text{and} $\forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow (TP, [SQ]) \in \text{TRel}$
\text{and} $\forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, [TQ]) \in \text{TRel}$
\text{(proof)}

\text{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 indRelSTEO, 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 $SRel : (\text{proc}S \times \text{proc}S)$ set

and $TRel : (\text{proc}T \times \text{proc}T)$ set

Assumes fullAbs: fully-abstract $SRel$ $TRel$

and reflS: refl $SRel$

and transT: trans $TRel$

Shows $\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}

(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 $SRel : (\text{proc}S \times \text{proc}S)$ set

and $TRel : (\text{proc}T \times \text{proc}T)$ set

Shows (fully-abstract $SRel$ $TRel$ \land refl $SRel$ \land trans $TRel$) =

(\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 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}

(proof)

9.4 Full Abstraction w.r.t. Equivalences

If there exists a relation $\text{Rel}$ 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 $\text{Rel} : (\text{proc}S, \text{proc}T) \text{Proc} \times (\text{proc}S, \text{proc}T) \text{Proc}$ set

Assumes enc: $\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}$

and trans: trans (symcl Rel)

Shows fully-abstract ((S1, S2). (SourceTerm S1, SourceTerm S2) \in symcl Rel)

\{(T1, T2). (TargetTerm T1, TargetTerm T2) \in symcl Rel\}

(proof)

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 $SRel : (\text{proc}S \times \text{proc}S)$ set

and $TRel : (\text{proc}T \times \text{proc}T)$ set

Assumes fullAbs: fully-abstract $SRel$ $TRel$

and reflT: refl $TRel$

and symmT: sym $TRel$

and transT: trans $TRel$

Shows $\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}$

\land SRel = ((S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{symcl Rel})

\land TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{symcl Rel}\}

\land \text{preorder} (\text{symcl Rel})

(proof)

Lemma (in encoding) fully-abstract-impl-symcl-source-target-relation-is-preorder:

Fixes $SRel : (\text{proc}S \times \text{proc}S)$ set

and $TRel : (\text{proc}T \times \text{proc}T)$ set
assumes fullAbs: fully-abstract \((\text{symcl } (SRel^=))^+\) \((\text{symcl } (TRel^=))^+\)
shows \(\exists \text{ Rel. } \forall S. \ (\text{SourceTerm } S, \text{TargetTerm } ([S]) \in \text{ Rel})\)
\(\land (\text{symcl } (SRel^=))^+ = \{(S1, S2). \ (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{ symcl Rel}\}\)
\(\land (\text{symcl } (TRel^=))^+ = \{(T1, T2). \ (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{ symcl Rel}\}\)
\(\land \ \text{ preorder } (\text{symcl Rel})\)
(proof)

lemma (in encoding) fully-abstract-wrt-preorders-impl-source-target-relation-is-trans:
fixes SRel :: \(\langle \text{procS } \times \text{procS} \rangle\) set
and TRel :: \(\langle \text{procT } \times \text{procT} \rangle\) set
assumes fullAbs: fully-abstract SRel TRel
shows \(\exists \text{ Rel. } \forall S. \ (\text{SourceTerm } S, \text{TargetTerm } ([S]) \in \text{ Rel})\)
\(\land \text{ SRel} = \{(S1, S2). \ (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{ Rel}\}\)
\(\land \text{ TRel} = \{(T1, T2). \ (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{ Rel}\}\)
\(\land \exists \text{ refl SRel } \land \text{ trans TRel}\)
\(\quad \text{symcl } (\text{Rel}) \quad \rightarrow \\
\quad \text{trans } (\text{Rel} \cup \{(P, Q). \ \exists S. \ ([S] \in T \ P \land S \in S \ Q)\})\)
(proof)

lemma (in encoding) fully-abstract-wrt-preorders-impl-source-target-relation-is-trans-B:
fixes SRel :: \(\langle \text{procS } \times \text{procS} \rangle\) set
and TRel :: \(\langle \text{procT } \times \text{procT} \rangle\) set
assumes fullAbs: fully-abstract SRel TRel
and reflT: refl TRel
and transT: trans TRel
shows \(\exists \text{ Rel. } \forall S. \ (\text{SourceTerm } S, \text{TargetTerm } ([S]) \in \text{ Rel})\)
\(\land \text{ SRel} = \{(S1, S2). \ (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{ symcl Rel}\}\)
\(\land \text{ TRel} = \{(T1, T2). \ (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{ symcl Rel}\}\)
\(\land \exists \text{ trans } (\text{Rel} \cup \{(P, Q). \ \exists S. \ ([S] \in T \ P \land S \in S \ Q)\})\)
(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 :: \(\langle \text{procS } \times \text{procS} \rangle\) set
and TRel :: \(\langle \text{procT } \times \text{procT} \rangle\) set
shows (fully-abstract SRel TRel \land \text{equivalence } TRel) =
\(\exists \text{ Rel. } \forall S. \ (\text{SourceTerm } S, \text{TargetTerm } ([S]) \in \text{ Rel})\)
\(\land \text{ SRel} = \{(S1, S2). \ (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{ symcl Rel}\}\)
\(\land \text{ TRel} = \{(T1, T2). \ (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{ symcl Rel}\}\)
\(\land \text{ preorder } (\text{symcl Rel})\)
(proof)

lemma (in encoding) fully-abstract-iff-symcl-source-target-relation-is-preorder:
fixes SRel :: \(\langle \text{procS } \times \text{procS} \rangle\) set
and TRel :: \(\langle \text{procT } \times \text{procT} \rangle\) set
shows fully-abstract (symcl (SRel^=))^+ \((\text{symcl } (TRel^=))^+\) =
\(\exists \text{ Rel. } \forall S. \ (\text{SourceTerm } S, \text{TargetTerm } ([S]) \in \text{ Rel})\)
\(\land \text{ symcl } (SRel^=)^+ = \{(S1, S2). \ (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{ symcl Rel}\}\)
\(\land \text{ symcl } (TRel^=)^+ = \{(T1, T2). \ (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{ symcl Rel}\}\)
\(\land \text{ preorder } (\text{symcl Rel})\)
(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 \( SRel :: (\langle \text{procS} \times \text{procS} \rangle) \) set
\[ \text{and } TRel :: (\langle \text{procT} \times \text{procT} \rangle) \] set
\[ \text{and } Rel :: (\langle \text{procS}, \text{procT} \rangle) \) \text{Proc} \times (\langle \text{procS}, \text{procT} \rangle) \) \text{Proc} \) set
\[ \text{assumes } \text{fullAbs: fully-abstract } SRel \) TRel \]
\[ \text{and } \text{rel: } Rel = ((\text{indRelSTEQ} \) SRel \) TRel) \]
\[ \text{∪ } \left\{ (P, Q). (P \in \text{ProcS} \) \land \) Q \in \text{ProcS}) \right\} \)
\[ \text{∪ } \left\{ (P, \text{procT}). \exists S P. SP \in S P \land \text{procS} \) Q \land \exists S \) \text{procS} \) (SP, SQ) \in SRel \}
\[ \vee (\exists TP \) TQ. TP \in T P \land \text{procT} \) TQ \in T Q \land (\text{TP}, \text{TQ}) \in TRel) \]
\[ \text{shows } \text{(refl SRel} \) \text{trans TRel) = tran } \text{Rel) \)} \]

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

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:**
\[ \text{fixes } Rel :: (\langle \text{procS}, \text{procT} \rangle) \) \text{Proc} \times (\langle \text{procS}, \text{procT} \rangle) \) \text{Proc} \) set
\[ \text{assumes } \text{enc: } \forall S. (\text{SourceTerm S, TargetTerm (\langle S \rangle)}) \in \text{Rel} \]
\[ \land \) (TargetTerm (\langle S \rangle, \text{SourceTerm S) \in \text{Rel} \}
\[ \text{and } \text{trans; trans Rel), \}) \]
\[ \text{shows fully-abstract } \{ (S1, S2). (\text{SourceTerm S1, SourceTerm S2) \in \text{Rel} \}
\[ \{ (T1, T2). (\text{TargetTerm T1, TargetTerm T2) \in \text{Rel} \}
\[ \langle \text{proof} \rangle \]

**lemma (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-is-transB:**
\[ \text{fixes } SRel :: (\langle \text{procS} \times \text{procS} \rangle) \] set
\[ \text{and } TRel :: (\langle \text{procT} \times \text{procT} \rangle) \] set
\[ \text{assumes } \text{procS: preorder } \text{TRel) \}
\[ \text{shows fully-abstract } SRel \) TRel \]
\[ \langle \text{proof} \rangle \]

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:**
\[ \text{fixes } SRel :: (\langle \text{procS} \times \text{procS} \rangle) \] set
\[ \text{and } TRel :: (\langle \text{procT} \times \text{procT} \rangle) \] set
\[ \text{shows } \text{fully-abstract } SRel \) TRel \) \text{refl SRel} \) \text{trans TRel) = \}
\[ \langle \text{proof} \rangle \]
10 Combining Criteria

So far we considered the effect of single criteria on encodings. Often the quality of an encoding is prescribed by a set of different criteria. In the following we analyse the combined effect of criteria. This way we can compare criteria as well as identify side effects that result from combinations of criteria. We start with some technical lemmata. To combine the effect of different criteria we combine the conditions they induce. If their effect can be described by a predicate on the pairs of the relation, as in the case of success sensitiveness or divergence reflection, combining the effects is simple.

**Lemma (in encoding) combine-conditions-on-pairs-of-relations:**
- **Fixes** \( \text{RelA, RelB} \) :: \( \langle \text{procS, procT} \rangle \) Proc \( \times \) \( \langle \text{procS, procT} \rangle \) Proc \( \Rightarrow \) bool
- **And** \( \text{CondA, CondB} \) :: \( \langle \text{procS, procT} \rangle \) Proc \( \times \) \( \langle \text{procS, procT} \rangle \) Proc \( \Rightarrow \) bool
- **Assumes** \( \forall (P, Q) \in \text{RelA. CondA} (P, Q) \)
- **And** \( \forall (P, Q) \in \text{RelB. CondB} (P, Q) \)
- **Shows** \( (\forall (P, Q) \in \text{RelA} \cap \text{RelB. CondA} (P, Q)) \land (\forall (P, Q) \in \text{RelA} \cap \text{RelB. CondB} (P, Q)) \)

**Lemma (in encoding) combine-conditions-on-sets-and-pairs-of-relations:**
- **Fixes** \( \text{RelA, RelB} \) :: \( \langle \text{procS, procT} \rangle \) Proc \( \times \) \( \langle \text{procS, procT} \rangle \) Proc \( \Rightarrow \) bool
- **And** \( \text{CondA, CondB} \) :: \( \langle \text{procS, procT} \rangle \) Proc \( \times \) \( \langle \text{procS, procT} \rangle \) Proc \( \Rightarrow \) bool
- **Assumes** \( \forall (P, Q) \in \text{RelA. CondA} (P, Q) \)
- **And** \( \forall (P, Q) \in \text{RelB. CondB} (P, Q) \)
- **And** \( \text{CondA} \land \text{Rel} \subseteq \text{RelA} \)
- **And** \( \text{CondB} \land \text{Rel} \subseteq \text{RelB} \)
- **Shows** \( (\forall (P, Q) \in \text{RelA. CondA} (P, Q)) \land (\forall (P, Q) \in \text{Rel. CondB} (P, Q)) \)

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.
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 proof \rangle

lemma (in encoding) $\text{indRelR-impl-cond-respects-predA-and-reflects-predB}$:

fixes $\text{PredA} \text{PredB} :: (\text{procS}, \text{procT}) \rightarrow \text{bool}$

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})$
\implies (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-reflects-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})$

\langle 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) $\text{WSS-DR-iff-source-target-rel}$:

fixes $\text{success} :: \text{barbs}$

shows $\text{enc-weakly-respects-barb-set } \{\text{success}\} \land \text{enc-reflects-divergence}$
\implies (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})$
\land (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-reflects-divergence Rel (STCalWB SWB TWB)}$\{success\}$
\land (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-reflects-divergence Rel (STCal Source Target)})$

\langle proof \rangle

lemma (in encoding-wrt-barbs) $\text{SS-DR-iff-source-target-rel}$:

fixes $\text{success} :: \text{barbs}$

shows $\text{enc-respects-barb-set } \{\text{success}\} \land \text{enc-reflects-divergence}$
\implies (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})$
\land (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-respects-barb-set Rel (STCalWB SWB TWB)}$\{success\}$
\land (\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{rel-reflects-divergence Rel (STCal Source Target)})$

\langle 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) $\text{enc-and-TRel-impl-indRelRTPO-weakly-respects-success}$:

fixes $\text{success} :: \text{barbs}$
and $\text{TRel} :: (\text{procT} \times \text{procT})$ set

assumes $\text{encRS}:: \text{enc-weakly-respects-barb-set } \{\text{success}\}$
\land $\text{trelPS}:: \text{rel-weakly-preserves-barb-set TRel TWB } \{\text{success}\}$
\land $\text{trelRS}:: \text{rel-weakly-reflects-barb-set TRel TWB } \{\text{success}\}$

shows $\text{rel-weakly-respects-barb-set (indRelRTPO TRel (STCalWB SWB TWB)}$\{success\}$

\langle proof \rangle

lemma (in encoding-wrt-barbs) $\text{enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs}$:

fixes $\text{TRel} :: (\text{procT} \times \text{procT})$ set

assumes $\text{encRS}:: \text{enc-weakly-respects-barbs}$
\land $\text{trelPS}:: \text{rel-weakly-preserves-barbs TRel TWB}$
\land $\text{trelRS}:: \text{rel-weakly-reflects-barbs TRel TWB}$

shows $\text{rel-weakly-respects-barbs (indRelRTPO TRel (STCalWB SWB TWB)}$
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 success :: 'barbs
  and 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)
\( \land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm} \ T_1, \text{TargetTerm} \ T_2) \in \text{Rel}\} \)
\( \land (\forall S \ T. (\text{SourceTerm} \ S, \text{TargetTerm} \ T) \in \text{Rel} \rightarrow ([S], T) \in T_{\text{Rel}}) \)
\( \land \text{weak-reduction-bisimulation Rel} (\text{STCal} \text{ Source Target})' \land \text{preorder Rel} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \)

(proof)

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

**fixes success :: 'bars**

**and** \( T_{\text{Rel}} := (\text{procT } \times \text{procT}) \text{ set} \)

**shows (operational-corresponding} T_{\text{Rel}} \land \text{preorder T}_{\text{Rel}} \land \text{weak-reduction-bisimulation} T_{\text{Rel}} \text{ Target} \)
\( \land \text{enc-weakly-respects-barbs} \land \text{rel-weakly-respects-barbs} T_{\text{Rel}} \text{ TWB} \)
\( \land \text{enc-weakly-respects-barb-set} \{\text{success}\} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB})' \land \text{preorder Rel} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \)

(proof)

An encoding is success sensitive and weakly operational corresponding w.r.t. a correspondence simulation \( T_{\text{Rel}} \) that respects success if there exists a correspondence simulation that includes \( T_{\text{Rel}} \) 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 :: 'bars**

**and** \( T_{\text{Rel}} := (\text{procT } \times \text{procT}) \text{ set} \)

**shows (weakly-operational-corresponding} T_{\text{Rel}} \land \text{preorder T}_{\text{Rel}} \land \text{weak-reduction-correspondence-simulation} T_{\text{Rel}} \text{ Target} \)
\( \land \text{enc-weakly-respects-barbs} \land \text{rel-weakly-respects-barbs} T_{\text{Rel}} \text{ TWB} \)
\( \land \text{rel-weakly-respects-barb-set} \{\text{success}\} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB})' \land \text{preorder Rel} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \)

(proof)

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

**fixes success :: 'bars**

**and** \( T_{\text{Rel}} := (\text{procT } \times \text{procT}) \text{ set} \)

**shows (weakly-operational-corresponding} T_{\text{Rel}} \land \text{preorder T}_{\text{Rel}} \land \text{weak-reduction-correspondence-simulation} T_{\text{Rel}} \text{ Target} \)
\( \land \text{enc-weakly-respects-barbs} \land \text{enc-weakly-respects-barb-set} \{\text{success}\} \)
\( \land \text{rel-weakly-respects-barbs} T_{\text{Rel}} \text{ TWB} \land \text{rel-weakly-respects-barb-set} T_{\text{Rel}} \text{ TWB} \{\text{success}\} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB})' \land \text{preorder Rel} \)
\( \land \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \)

(proof)

An encoding is strongly success sensitive and strongly operational corresponding w.r.t. a strong bisimulation \( T_{\text{Rel}} \) that strongly respects success if there exists a strong bisimulation that includes \( T_{\text{Rel}} \) 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 :: 'bars**
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:
  fixes success :: 'barbs
  and TRel :: ('procT × 'procT) set
  shows (strongly-operational-corresponding TRel ∧ preorder TRel
  ∧ strong-reduction-bisimulation TRel Target
  ∧ enc-respects-barb-set {success} ∧ rel-respects-barb-set TRel TWB {success})
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  ∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
  ∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success})

(proof)

lemma (in encoding-wrt-barbs) OC-SS-DR-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})
  ∧ rel-reflects-divergence Rel (STCal Source Target))

(proof)

lemma (in encoding-wrt-barbs) WOC-SS-DR-wrt-preorder-iff-source-target-rel:
  fixes success :: 'barbs
  and TRel :: ('procT × 'procT) set
  shows (weakly-operational-corresponding TRel ∧ preorder TRel
  ∧ weak-reduction-correspondence-simulation TRel Target
  ∧ enc-weakly-respects-barb-set {success})
  ∧ rel-weakly-respects-barb-set TRel TWB {success}
  ∧ enc-reflects-divergence ∧ rel-reflects-divergence TRel Target)
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  ∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}

(proof)
lemma (in encoding-wrt-barbs) OC-SS-DR-wrt-preorder-iff-source-target-rel:
  \begin{align*}
  \text{fixes} & \; SRel :: \text{ 'bars} \\
  \text{and} & \; TRel :: \langle \text{ 'procT} \times \text{ 'procT} \rangle \text{ set} \\
  \text{shows} & \; \langle \text{operational-corresponding} \ TRel \land \text{ preorder} \ TRel \land \text{ weak-reduction-bisimulation} \ TRel \ TTarget \rangle \\
  \text{and} & \; \langle \text{enc-weakly-respects-barb-set} \ SRel \ TRel \rangle \text{ TWB} \{ \text{success} \} \\
  \text{and} & \; \langle \text{enc-reflects-divergence} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
  \text{and} & \; \langle \text{rel-reflects-divergence} \ TRel \ TTarget \rangle \\
  \langle & \; (\exists \text{ Rel} \cdot (\forall S. (\langle \text{SourceTerm} \ S, \text{TargetTerm} \ ([S]) \rangle) \in \text{ Rel}) \\
  \text{and} & \; TRel = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{ Rel} \} \\
  \text{and} & \; (\forall S T. (\langle \text{SourceTerm} \ S, \text{TargetTerm} \ T \rangle) \in \text{ Rel} \longrightarrow ([S], T) \in \text{ TRel}) \\
  \text{and} & \; \langle \text{weak-reduction-bisimulation} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
  \text{and} & \; \langle \text{rel-reflects-divergence} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
  \text{and} & \; \langle \text{rel-reflects-divergence} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
  \text{(proof)} \end{align*}

\begin{align*}
\text{lemma (in encoding-wrt-barbs) SOC-SS-DR-wrt-preorder-iff-source-target-rel:} \\
\text{fixes} & \; SRel :: \text{ 'bars} \\
\text{and} & \; TRel :: \langle \text{ 'procT} \times \text{ 'procT} \rangle \text{ set} \\
\text{shows} & \; \langle \text{strongly-operational-corresponding} \ TRel \land \text{ preorder} \ TRel \rangle \\
\text{and} & \; \langle \text{weak-reduction-bisimulation} \ TRel \ TTarget \rangle \\
\text{and} & \; \langle \text{enc-respects-barb-set} \ SRel \ TRel \rangle \text{ TWB} \{ \text{success} \} \\
\text{and} & \; \langle \text{enc-reflects-divergence} \ TRel \ TTarget \rangle \\
\langle & \; (\exists \text{ Rel} \cdot (\forall S. (\langle \text{SourceTerm} \ S, \text{TargetTerm} \ ([S]) \rangle) \in \text{ Rel}) \\
\text{and} & \; TRel = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{ Rel} \} \\
\text{and} & \; (\forall S T. (\langle \text{SourceTerm} \ S, \text{TargetTerm} \ T \rangle) \in \text{ Rel} \longrightarrow ([S], T) \in \text{ TRel}) \\
\text{and} & \; \langle \text{weak-reduction-bisimulation} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
\text{and} & \; \langle \text{rel-reflects-barb-set} \ SRel \ TRel \rangle \text{ TWB} \{ \text{success} \} \\
\text{and} & \; \langle \text{rel-reflects-divergence} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
\text{and} & \; \langle \text{rel-reflects-divergence} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
\text{(proof)} \end{align*}

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.

\begin{align*}
\text{lemma (in encoding) FS-OC-modulo-equivalences-iff-source-target-relation:} \\
\text{fixes} & \; SRel :: \langle \text{ 'procS} \times \text{ 'procS} \rangle \text{ set} \\
\text{and} & \; TRel :: \langle \text{ 'procT} \times \text{ 'procT} \rangle \text{ set} \\
\text{assumes} & \; \langle \text{eqS: equivalence} \ SRel \rangle \\
\text{and} & \; \langle \text{eqT: equivalence} \ TRel \rangle \\
\text{shows} & \; \langle \text{fully-abstract} \ SRel \ TRel \rangle \\
\text{and} & \; \langle \text{operational-corresponding} \ TRel \land \text{ weak-reduction-bisimulation} \ TRel \ TTarget \rangle \\
\text{iff} & \; (\exists \text{ Rel}. (\forall S. (\langle \text{SourceTerm} \ S, \text{TargetTerm} \ ([S]) \rangle) \in \text{ Rel}) \\
\text{and} & \; \langle \text{SRel} = \{(S1, S2), (\text{SourceTerm} \ S1, \text{SourceTerm} \ S2) \in \text{ Rel} \} \\
\text{and} & \; \langle \text{TRel} = \{(T1, T2), (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{ Rel} \} \\
\text{and} & \; \langle \text{trans Rel} \land \text{ weak-reduction-bisimulation} \ TRel \ TTarget \rangle \text{ TRel} \{ \text{success} \} \\
\text{(proof)} \end{align*}

\begin{align*}
\text{lemma (in encoding) FA-SOC-modulo-equivalences-iff-source-target-relation:} \\
\text{fixes} & \; SRel :: \langle \text{ 'procS} \times \text{ 'procS} \rangle \text{ set} \\
\text{and} & \; TRel :: \langle \text{ 'procT} \times \text{ 'procT} \rangle \text{ set} \\
\text{assumes} & \; \langle \text{eqS: equivalence} \ SRel \rangle 
\end{align*}
and $eqT$: equivalence $TRel$

shows fully-abstract $SRel$ $TRel$ ∧ strongly-operational-corresponding $TRel$
∧ strong-reduction-bisimulation $TRel$ Target $\leftrightarrow$ ($\exists$ $Rel$.
(∀ $S$. (SourceTerm $S$, TargetTerm $([S])$) ∈ $Rel$ ∧ (TargetTerm $([S])$, SourceTerm $S$) ∈ $Rel$)
∧ $SRel$ = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ $Rel$}
∧ $TRel$ = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ $Rel$} ∧ trans $Rel$
∧ strong-reduction-bisimulation $Rel$ (STCal Source Target))

 ⟨proof⟩

An encoding that is fully abstract w.r.t. the equivalences $SRel$ and $TRel$ and operationally corresponding w.r.t. $TRel$ ensures that $SRel$ is a bisimulation iff $TRel$ is a bisimulation.

lemma (in encoding) FA-and-OC-and-$TRel$-impl-$SRel$-bisimulation:

fixes $SRel$ :: ('procS × 'procS) set
and $TRel$ :: ('procT × 'procT) set
assumes fullAbs: fully-abstract $SRel$ $TRel$
and opCom: operational-complete $TRel$
and opSou: operational-sound $TRel$
and symmT: sym $TRel$
and transT: trans $TRel$
and bisimT: weak-reduction-bisimulation $TRel$ Target

shows weak-reduction-bisimulation $SRel$ Source

 ⟨proof⟩

lemma (in encoding) FA-and-SOC-and-$TRel$-impl-$SRel$-strong-bisimulation:

fixes $SRel$ :: ('procS × 'procS) set
and $TRel$ :: ('procT × 'procT) set
assumes fullAbs: fully-abstract $SRel$ $TRel$
and opCom: strongly-operational-complete $TRel$
and opSou: strongly-operational-sound $TRel$
and symmT: sym $TRel$
and transT: trans $TRel$
and bisimT: strong-reduction-bisimulation $TRel$ Target

shows strong-reduction-bisimulation $SRel$ Source

 ⟨proof⟩

lemma (in encoding) FA-and-OC-impl-$SRel$-iff-$TRel$-bisimulation:

fixes $SRel$ :: ('procS × 'procS) set
and $TRel$ :: ('procT × 'procT) set
assumes fullAbs: fully-abstract $SRel$ $TRel$
and opCor: operational-corresponding $TRel$
and symmT: sym $TRel$
and transT: trans $TRel$
and sury: ∀ T. ∃ S. T = $[S]$

shows weak-reduction-bisimulation $SRel$ Source $\leftrightarrow$ weak-reduction-bisimulation $TRel$ Target

 ⟨proof⟩

end