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

Kirstin Peters
TU Dresden, Germany

Rob van Glabbeek
NICTA†, Sydney, Australia
Computer Science and Engineering, UNSW, Sydney, Australia

August 05, 2015

Abstract

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

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

---

*Supported by funding of the Excellence Initiative by the German Federal and State Governments (Institutional Strategy, measure ‘support the best’).
†NICTA is funded by the Australian Government through the Department of Communications and the Australian Research Council through the ICT Centre of Excellence Program.
Contents

1 Relations 3
  1.1 Basic Conditions 3
  1.2 Preservation, Reflection, and Respection of Predicates 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 ∩ Rel<sup>−1</sup>) = (symcl Rel)<sup>−1</sup>
  ⟨proof⟩

The symmetric closure of a reflexive relation is reflexive.

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

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

lemma refl-symm-trans-closure-is-symm-refl-trans-closure:
The reflexive closure of a symmetric relation is symmetric.

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

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

**Assumes** \( \text{sym Rel} \)

**Shows** \( \text{sym} (\text{Rel}^\text{refl}) \)

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

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

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

**Assumes** \( \text{refl Rel} \)

**Shows** \( \text{Rel}^\text{refl} = \text{Rel} \)

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

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

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

**Assumes** \( \text{sym Rel} \)

**Shows** \( \text{symcl Rel} = \text{Rel} \)

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

**Lemma rtrancl-of-preorder:**

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

**Assumes** \( \text{preorder Rel} \)

**Shows** \( \text{Rel}^\text{rtrancl} = \text{Rel} \)

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

**Lemma refl-trans-closure-subset-of-refl-symm-trans-closure:**

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

**Shows** \( \text{Rel}^\text{refl} \subseteq (\text{symcl} (\text{Rel}^\text{refl}))^\text{rtrancl} \)

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

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

**Assumes** \( \text{condB} : \forall a \ b \ c. \ (a, b) \in \text{Rel} \land (a, c) \in \text{Rel} \land (b, c) /\in \text{Rel} \rightarrow (b, a) \in \text{Rel} \lor (c, a) \in \text{Rel} \land (a, c) \notin \text{Rel} \)

**Assumes** \( \text{reflR} : \text{refl Rel} \)

**Assumes** \( \text{tranR} : \text{trans Rel} \)

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

1.2 Preservation, Reflection, and Respection of Predicates

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

**Abbreviation** \( \text{rel-preserves-pred} :: (a \times a) \text{ set} \Rightarrow (a \Rightarrow \text{bool}) \Rightarrow \text{bool} \)

**Abbreviation** \( \text{rel-preserves-pred} \text{ Rel} \text{ Pred} \equiv \forall a \ b. \ (a, b) \in \text{Rel} \land \text{Pred} \ a \rightarrow \text{Pred} \ b \)
A relation \( R \) reflects some predicate \( P \) if \( P(b) \) implies \( P(a) \) for all \((a, b)\) in \( R \).

**abbreviation** rel-reflects-pred :: \( (\mathcal{P} \times \mathcal{P}) \rightarrow \mathbb{B} \)

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

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

**abbreviation** rel-respects-pred :: \( (\mathcal{P} \times \mathcal{P}) \rightarrow \mathbb{B} \)

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

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

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

fixes \( \text{Rel} : (\mathcal{P} \times \mathcal{P}) \rightarrow \mathbb{B} \)
and \( \text{Pred} : \mathcal{P} \rightarrow \mathbb{B} \)
assumes symm: sym \( \text{Rel} \)
shows rel-preserves-pred \( \text{Rel} \: \text{Pred} = \text{rel-reflects-pred} \: \text{Rel} \: \text{Pred} \)
and rel-preserves-pred \( \text{Rel} \: \text{Pred} = \text{rel-respects-pred} \: \text{Rel} \: \text{Pred} \)
and 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:

fixes \( \text{Rel} : (\mathcal{P} \times \mathcal{P}) \rightarrow (\mathcal{P} \rightarrow \mathbb{B}) \)
and \( \text{Pred} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathbb{B} \)
assumes symm: sym \( \text{Rel} \)
shows rel-preserves-binary-pred \( \text{Rel} \: \text{Pred} = \text{rel-reflects-binary-pred} \: \text{Rel} \: \text{Pred} \)
and rel-preserves-binary-pred \( \text{Rel} \: \text{Pred} = \text{rel-respects-binary-pred} \: \text{Rel} \: \text{Pred} \)
and 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:

fixes \( \text{Rel} : (\mathcal{P} \times \mathcal{P}) \rightarrow \mathbb{B} \)
and \( \text{Pred} : \mathcal{P} \rightarrow \mathbb{B} \)
assumes preservation: rel-preserves-pred \( \text{Rel} \: \text{Pred} \)
shows rel-preserves-pred \( \text{Rel}^- \: \text{Pred} \)
and rel-preserves-pred \( \text{Rel}^+ \: \text{Pred} \)
and rel-preserves-pred \( \text{Rel}^* \: \text{Pred} \)

⟨proof⟩

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

fixes \( \text{Rel} : (\mathcal{P} \times \mathcal{P}) \rightarrow (\mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathbb{B}) \)
and \( \text{Pred} : \mathcal{P} \rightarrow \mathcal{P} \rightarrow \mathbb{B} \)
assumes preservation: rel-preserves-binary-pred \( \text{Rel} \: \text{Pred} \)
shows rel-preserves-binary-pred \( \text{Rel}^- \: \text{Pred} \)
and rel-preserves-binary-pred \( \text{Rel}^+ \: \text{Pred} \)
and 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:
2 Process Calculi

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

2.1 Reduction Semantics

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

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

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

```ml
(- ‡ - [70, 70, 70] 80)
```

```ml
where
P ↦→ Cal Q ≡ Reductions Cal P Q
```

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

```ml
primrec nSteps :: ′proc ⇒ ′proc processCalculus ⇒ nat ⇒ ′proc ⇒ bool
(- ‡ - ‡ [70, 70, 70] 80)
```

```ml
where
P ↦→ Cal^0 Q = (P = Q) |
P ↦→ Cal^{Suc n} Q = (∃ P'. P ↦→ Cal^n P' ∧ P' ↦→ Cal Q)
```

**definition steps :: ′proc ⇒ ′proc processCalculus ⇒ ′proc ⇒ bool**

```ml
(- ‡ - ∗ [70, 70, 70] 80)
```

```ml
where
P ↦→ Cal^∗ Q ≡ ∃ n. P ↦→ Cal^n Q
```

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

**definition divergent :: ′proc ⇒ ′proc processCalculus ⇒ ′proc ⇒ bool**

```ml
(- ‡ - ω [70, 70, 70] 80)
```

```ml
where
P ↦→ (Cal^ω) ≡ ∀ P'. P ↦→ Cal^∗ P' → (∃ P''. P' ↦→ Cal P'')
```

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

**lemma steps-refl**:  
fixes Cal :: ′proc processCalculus  
and P :: ′proc  
shows P ↦→ Cal^∗ P  
(proof)

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

**lemma step-to-steps**:  
fixes Cal :: ′proc processCalculus  
and P P' :: ′proc  
assumes step :: P ↦→ Cal P'  
shows P ↦→ 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 ∀ P Q R. P ↦→ Cal^{n1} Q ∧ Q ↦→ Cal^{n2} R → P ↦→ Cal^{(n1 + n2)} R  
(proof)

**lemma steps-add**:  
fixes Cal :: ′proc processCalculus  
and P Q R :: ′proc  
assumes A1: P ↦→ Cal^∗ Q  
and A2: Q ↦→ Cal^∗ R  
shows P ↦→ 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
  (∧[70, 70] 80)

where
  P↓<CWB>a ⇒ HasBarb CWB P a

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

abbreviation reachesBarb :: (′proc, ′barbs) calculusWithBarbs ⇒ ′barbs ⇒ bool
  (∨<70<70<70<80)

where
  P⇓<CWB>a ∈ ∃P'. P'→(Calculus CWB)* P ∧ P↓<CWB>a

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

abbreviation rel-preserves-barb-set :: (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ ′barbs set ⇒ bool
  (−<70<70<70<80)

where
  rel-preserves-barb-set Rel CWB Barbs ≡ rel-preserves-binary-pred Rel (λP a. a ∈ Barbs ∧ P↓<CWB>a)

abbreviation rel-preserves-barbs :: (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ bool
  (∨<70<70<70<80)

where
  rel-preserves-barbs Rel CWB ≡ rel-preserves-binary-pred Rel (HasBarb CWB)

lemma preservation-of-barbs-and-set-of-barbs:
  fixes Rel :: (′proc × ′proc) set
  and CWB :: (′proc, ′barbs) calculusWithBarbs
  shows rel-preserves-barbs Rel CWB = (∀ Barbs. rel-preserves-barb-set Rel CWB Barbs)
  ⟨proof⟩

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

abbreviation rel-reflects-barb-set :: (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ ′barbs set ⇒ bool
  (−<70<70<70<80)

where
  rel-reflects-barb-set Rel CWB Barbs ≡ rel-reflects-binary-pred Rel (HasBarb CWB)

abbreviation rel-reflects-barbs :: (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ bool
  (∨<70<70<70<80)

where
  rel-reflects-barbs Rel CWB ≡ rel-reflects-binary-pred Rel (HasBarb CWB)

lemma reflection-of-barbs-and-set-of-barbs:
  fixes Rel :: (′proc × ′proc) set
  and CWB :: (′proc, ′barbs) calculusWithBarbs
  shows rel-reflects-barbs Rel CWB = (∀ Barbs. rel-reflects-barb-set Rel CWB Barbs)
  ⟨proof⟩

A relation respects barbs if it preserves and reflects barbs.

abbreviation rel-respects-barb-set
```
rel-respects-barb-set Rel CWB Barbs \equiv
rel-preserves-barb-set Rel CWB Barbs \land rel-reflects-barb-set Rel CWB Barbs

abbreviation rel-respects-bars
\quad :: \; (′proc × ′proc) set \Rightarrow \; (′proc, ′barbs) calculusWithBarbs \Rightarrow \; ′barbs set \Rightarrow \; bool
where
rel-respects-bars Rel CWB \equiv
rel-preserves-bars Rel CWB \land rel-reflects-bars Rel CWB

lemma respection-of-bars-and-set-of-bars:
\quad \text{fixes} \quad Rel :: \; (′proc × ′proc) set
\quad \text{and} \quad CWB :: \; (′proc, ′barbs) calculusWithBarbs
\quad \text{shows} \quad rel-respects-bars Rel CWB = (∀ Barbs. \; rel-respects-barb-set Rel CWB Barbs)
\quad \langle proof \rangle

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

lemma preservation-of-bars-and-closures:
\quad \text{fixes} \quad Rel :: \; (′proc × ′proc) set
\quad \text{and} \quad CWB :: \; (′proc, ′barbs) calculusWithBarbs
\quad \text{assumes} \quad preservation: \; rel-preserves-bars Rel CWB
\quad \text{shows} \quad rel-preserves-bars (Rel symcl) CWB
\quad \text{and} \quad rel-preserves-bars (Rel +) CWB
\quad \text{and} \quad rel-preserves-bars (Rel ∗) CWB
\quad \langle proof \rangle

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

lemma reflection-of-bars-and-closures:
\quad \text{fixes} \quad Rel :: \; (′proc × ′proc) set
\quad \text{and} \quad CWB :: \; (′proc, ′barbs) calculusWithBarbs
\quad \text{assumes} \quad reflection: \; rel-reflects-bars Rel CWB
\quad \text{shows} \quad rel-reflects-bars (Rel symcl) CWB
\quad \text{and} \quad rel-reflects-bars (Rel +) CWB
\quad \text{and} \quad rel-reflects-bars (Rel ∗) CWB
\quad \langle proof \rangle

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

lemma respection-of-bars-and-closures:
\quad \text{fixes} \quad Rel :: \; (′proc × ′proc) set
\quad \text{and} \quad CWB :: \; (′proc, ′barbs) calculusWithBarbs
\quad \text{assumes} \quad respection: \; rel-respects-bars Rel CWB
\quad \text{shows} \quad rel-respects-bars (Rel symcl) CWB
\quad \text{and} \quad rel-respects-bars (symcl Rel) CWB
\quad \text{and} \quad rel-respects-bars (Rel +) CWB
\quad \text{and} \quad rel-respects-bars (symcl (Rel symcl)) CWB
\quad \text{and} \quad rel-respects-bars (symcl (Rel symcl)) + CWB
\quad \langle proof \rangle

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

abbreviation rel-weakly-preserves-barb-set
\quad :: \; (′proc × ′proc) set \Rightarrow \; (′proc, ′barbs) calculusWithBarbs \Rightarrow \; ′barbs set \Rightarrow \; bool
where
rel-weakly-preserves-barb-set Rel CWB Barbs \equiv
rel-preserves-binary-pred Rel (λ P a. \; a ∈ Barbs \land P << CWB a)

abbreviation rel-weakly-preserves-bars
\quad :: \; (′proc × ′proc) set \Rightarrow \; (′proc, ′barbs) calculusWithBarbs \Rightarrow \; bool
where
rel-weakly-preserves-barbs $\text{Rel} \ 	ext{CWB} \equiv \text{rel-preserves-binary-pred} \text{Rel} (\lambda P \ a. \ P \downarrow <\text{CWB}>a)$

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

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

**And** CWB $:: \ (\text{proc}, \text{bars} ) \text{calculusWithBarbs}$

** Shows** rel-weakly-preserves-barbs $\text{Rel} \ 	ext{CWB}$

= ($\forall \text{Barbs}. \ \text{rel-weakly-preserves-barb-set} \text{Rel} \ 	ext{CWB} \ 	ext{Barbs}$)

⟨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

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

where

rel-weakly-reflects-barb-set $\text{Rel} \ 	ext{CWB} \ 	ext{Barbs} \equiv$

rel-reflects-binary-pred $\text{Rel} (\lambda P \ a. \ a \in \text{Barbs} \land P \downarrow <\text{CWB}>a)$

**Abbreviation** rel-weakly-reflection-of-barbs

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

where

rel-weakly-reflected-bars $\text{Rel} \ 	ext{CWB} \equiv \text{rel-reflects-binary-pred} (\lambda P \ a. \ P \downarrow <\text{CWB}>a)$

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

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

**And** CWB $:: \ (\text{proc}, \text{bars} ) \text{calculusWithBarbs}$

** Shows** rel-weakly-reflected-bars $\text{Rel} \ 	ext{CWB} = (\forall \text{Barbs}. \ \text{rel-weakly-reflected-bars-set} \text{Rel} \ 	ext{CWB} \ 	ext{Barbs})$

⟨proof⟩

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

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

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

where

rel-weakly-respects-barb-set $\text{Rel} \ 	ext{CWB} \ 	ext{Barbs} \equiv$

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

and

rel-weakly-reflects-barb-set $\text{Rel} \ 	ext{CWB} \ 	ext{Barbs}$

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

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

where

rel-weakly-respects-barbs $\text{Rel} \ 	ext{CWB} \equiv$

rel-weakly-preserves-barbs $\text{Rel} \ 	ext{CWB}$

and

rel-weakly-reflects-barbs $\text{Rel} \ 	ext{CWB}$

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

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

**And** CWB $:: \ (\text{proc}, \text{bars} ) \text{calculusWithBarbs}$

** Shows** rel-weakly-resepecting-barbs $\text{Rel} \ 	ext{CWB} = (\forall \text{Barbs}. \ \text{rel-weakly-respecting-barb-set} \text{Rel} \ 	ext{CWB} \ 	ext{Barbs})$

⟨proof⟩

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

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

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

**And** CWB $:: \ (\text{proc}, \text{bars} ) \text{calculusWithBarbs}$

**Assumes** preservation: rel-weakly-preserves-barbs $\text{Rel} \ 	ext{CWB}$

** Shows** rel-weakly-preserves-barbs $\text{Rel} (\text{refl}) \ 	ext{CWB}$

and

rel-weakly-preserves-barbs $\text{Rel} (\text{trans}) \ 	ext{CWB}$

and

rel-weakly-preserves-barbs $\text{Rel} (\text{refl}) \ 	ext{CWB}$

⟨proof⟩

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

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

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

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

- **Fixes** \( \mathcal{R} \) :: \( \text{'proc} \times \text{'proc} \) set
- and \( \text{CWB} \) :: \( \text{'proc}, \text{'barbs} \) calculusWithBarbs
- **Assumes** respection: rel-weakly-respects-barbs \( \mathcal{R} \) \( \text{CWB} \)
- shows rel-weakly-respects-barbs \( (\text{rel-weakly-preserves-barbs (\mathcal{R}\equiv\text{CWB}}) \)
- and rel-weakly-preserves-barbs \( (\text{symcl (\mathcal{R}\equiv\text{CWB}}) \)
- and rel-weakly-preserves-barbs \( (\text{symcl (\mathcal{R}\equiv\text{CWB}}) \)
- and rel-weakly-preserves-barbs \( ((\text{symcl (\mathcal{R}\equiv\text{CWB}})\equiv\) CWB

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

\end{proof}

\end{theory}
lemma weak-barbed-simulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: weak-barbed-simulation Rel CWB
  shows weak-barbed-simulation (Rel=) CWB
  and weak-barbed-simulation (Rel+) CWB
  and weak-barbed-simulation (Rel*) CWB
⟨proof⟩

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

abbreviation weak-barbed-preservation-cond :: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool
  where
  weak-barbed-preservation-cond Rel CWB ≡ ∀ P Q a. (P, Q) ∈ Rel ∧ P↓CWB>P a −→ Q⇓CWB>Q

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 (Calculus 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
  ⟨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
  ⟨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 \((\text{Rel}^+)^*\) \(CWB\)
and weak-barbed-contrasimulation \((\text{Rel}^*)^*\) \(CWB\)

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

### 3.3 Coupled Simulation

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

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

\[\text{weak-reduction-coupled-simulation}:: (\text{'proc} \times \text{'proc}) \Rightarrow \text{bool}\]

where

\[\text{weak-reduction-coupled-simulation} \text{\ Rel Cal} \equiv \forall P Q P', (P, Q) \in \text{Rel} \land P \rightarrow_{\text{Cal}^+} P' \rightarrow (\exists Q'. Q \rightarrow_{\text{Cal}^+} Q' \land (P', Q') \in \text{Rel}) \land (\exists Q'. Q \rightarrow_{\text{Cal}^*} Q' \land (Q', P') \in \text{Rel})\]

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

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

\[\text{weak-barbed-coupled-simulation}:: (\text{'proc} \times \text{'proc}) \Rightarrow (\text{'proc, 'barbs}) \text{calculusWithBarbs} \Rightarrow \text{bool}\]

where

\[\text{weak-barbed-coupled-simulation} \text{\ Rel CWB} \equiv \text{weak-reduction-coupled-simulation} \text{\ (Calculus CWB)} \land \text{rel-weakly-preserves-barbs} \text{\ Rel CWB}\]

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

and Cal :: \text{'proc processCalculus}

shows weak-reduction-coupled-simulation \text{\ Rel Cal} = (weak-reduction-simulation \text{\ Rel Cal} \land weak-reduction-contrasimulation \text{\ Rel Cal})

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

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

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

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

shows weak-barbed-coupled-simulation \text{\ Rel CWB} = (weak-barbed-simulation \text{\ Rel CWB} \land weak-barbed-contrasimulation \text{\ Rel CWB})

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

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

and Cal :: \text{'proc processCalculus}

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

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

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

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

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

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

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

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

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

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

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

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

where

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

$\equiv$

$(\forall P \; Q \; P'. \; (P, Q) \in \text{Rel} \land P \mapsto \text{cal} \; P' \mapsto (\exists Q'. \; Q \mapsto \text{cal} \; Q' \land (P', Q') \in \text{Rel}))$

$\land (\forall P \; Q \; Q'. \; (P, Q) \in \text{Rel} \land Q \mapsto \text{cal} \; Q' \mapsto (\exists P'' \; Q'' \; P \mapsto \text{cal} \; P'' \land Q' \mapsto \text{cal} \; Q'' \land (P'', Q'') \in \text{Rel}))$

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

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

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

where

weak-barbed-correspondence-simulation $\Rightarrow$ Rel CWB

$$\equiv$$

weak-reduction-correspondence-simulation $\Rightarrow$ Rel (Calculus CWB)

$\land \text{ rel-weakly-respects-barbs} \Rightarrow$ Rel CWB

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

**inductive-set** cSim-cs $:: \langle \text{proc} \times \text{proc} \rangle \text{ set } \Rightarrow \langle \text{proc processCalculus} \Rightarrow \langle \text{proc} \times \text{proc} \rangle \text{ set} \rangle$

for Rel $:: \langle \text{proc} \times \text{proc} \rangle \text{ set}$

and Cal $:: \langle \text{proc processCalculus} \rangle$

where

left: $\llbracket Q \mapsto \text{cal} \; Q' \; (P', Q') \in \text{Rel} \rrbracket \Rightarrow (P', Q) \in \text{cSim-cs} \Rightarrow \text{Cal} \land$

right: $\llbracket P \mapsto \text{cal} \; P' \; (Q, P) \in \text{Rel} \rrbracket \Rightarrow (P', Q) \in \text{cSim-cs} \Rightarrow \text{Cal} \land$

trans: $\llbracket (P, Q) \in \text{cSim-cs} \Rightarrow \text{Cal} \; (Q, R) \in \text{cSim-cs} \Rightarrow \text{Cal} \rrbracket \Rightarrow (P, R) \in \text{cSim-cs} \Rightarrow \text{Cal}$

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

fixes Rel $:: \langle \text{proc} \times \text{proc} \rangle \text{ set}$

and Cal $:: \langle \text{proc processCalculus} \rangle$

assumes corrSim $::$ weak-reduction-correspondence-simulation $\Rightarrow$ Rel $\Rightarrow$ Cal

shows weak-reduction-coupled-simulation (cSim-cs Rel Cal) Cal

and $\forall P \; Q. \; (P, Q) \in \text{Rel} \mapsto (P, Q) \in \text{cSim-cs} \Rightarrow \text{Rel} \land (Q, P) \in \text{cSim-cs} \Rightarrow \text{Rel}$

(proof)

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

fixes Rel $:: \langle \text{proc} \times \text{proc} \rangle \text{ set}$

and CWB $:: \langle \text{proc, barbs} \rangle \text{ calculusWithBarbs}$

assumes corrSim $::$ weak-barbed-correspondence-simulation $\Rightarrow$ Rel CWB

shows weak-barbed-coupled-simulation (cSim-cs Rel (Calculus CWB)) CWB

and $\forall P \; Q. \; (P, Q) \in \text{Rel} \mapsto (P, Q) \in \text{cSim-cs} \Rightarrow \text{Rel} (\text{Calculus CWB})$

$\land (Q, P) \in \text{cSim-cs} \Rightarrow \text{Rel} (\text{Calculus CWB})$

(proof)

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

fixes Cal $:: \langle \text{proc processCalculus} \rangle$

and P Q R $:: \text{proc}$

and Rel $:: \langle \text{proc} \times \text{proc} \rangle \text{ set}$

assumes A1: $\forall Q'. \; Q \mapsto \text{cal} \; Q' \mapsto (\exists P'' \; Q''. \; P \mapsto \text{cal} \; P'' \land Q' \mapsto \text{cal} \; Q'' \land (P'', Q'') \in \text{Rel})$

and A2: $\forall R'. \; R \mapsto \text{cal} \; R' \mapsto (\exists Q'' \; R''. \; Q \mapsto \text{cal} \; Q'' \land R' \mapsto \text{cal} \; R'' \land (Q'', R'') \in \text{Rel})$

and A3: weak-reduction-simulation $\Rightarrow$ Rel Cal

and A4: trans Rel

shows $\forall R'. \; R \mapsto \text{cal} \; R' \mapsto (\exists P'' \; R''. \; P \mapsto \text{cal} \; P'' \land R' \mapsto \text{cal} \; R'' \land (P'', R'') \in \text{Rel})$
The reflexive and/or transitive closure of a weak correspondence simulation is a weak correspondence simulation.

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

fixes $\text{Rel} :: (\text{proc} \times \text{proc})$ set and $\text{Cal} :: \text{proc processCalculus}$

assumes $\text{corrSim} :: \text{weak-reduction-correspondence-simulation} \text{ Rel Cal}$

shows $\text{weak-reduction-correspondence-simulation} (\text{Rel}^=) \text{ Cal}$ and $\text{weak-reduction-correspondence-simulation} (\text{Rel}^+) \text{ Cal}$ and $\text{weak-reduction-correspondence-simulation} (\text{Rel}^\ast) \text{ Cal}$

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

fixes $\text{Rel} :: (\text{proc} \times \text{proc})$ set and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs

assumes $\text{corrSim} :: \text{weak-barbed-correspondence-simulation} \text{ Rel CWB}$

shows $\text{weak-barbed-correspondence-simulation} (\text{Rel}^=) \text{ CWB}$ and $\text{weak-barbed-correspondence-simulation} (\text{Rel}^+) \text{ CWB}$ and $\text{weak-barbed-correspondence-simulation} (\text{Rel}^\ast) \text{ CWB}$

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**

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

where

$\text{weak-reduction-bisimulation} \text{ Rel Cal} \equiv \forall P Q P'. (P, Q) \in \text{Rel} \land P \rightarrow \text{Cal} \ast P' \rightarrow (\exists Q'. Q \rightarrow \text{Cal} \ast Q' \land (P', Q') \in \text{Rel}))$

$\land (\forall P Q Q'. (P, Q) \in \text{Rel} \land Q \rightarrow \text{Cal} \ast Q' \rightarrow (\exists P'. P \rightarrow \text{Cal} \ast P' \land (P', Q') \in \text{Rel}))$

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

**Abbreviation weak-barbed-bisimulation**

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

where

$\text{weak-barbed-bisimulation} \text{ Rel CWB} \equiv \text{weak-reduction-bisimulation} \text{ Rel CWB} \land \text{rel-weakly-respects-barbs} \text{ Rel CWB}$

A symmetric weak simulation is a weak bisimulation.

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

fixes $\text{Rel} :: (\text{proc} \times \text{proc})$ set and $\text{Cal} :: \text{proc processCalculus}$

assumes $\text{sym Rel}$ and $\text{weak-reduction-simulation} \text{ Rel Cal}$

shows $\text{weak-reduction-bisimulation} \text{ Rel Cal}$

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

fixes $\text{Rel} :: (\text{proc} \times \text{proc})$ set and $\text{CWB} :: (\text{proc}, \text{barbs})$ calculusWithBarbs

assumes $\text{sym Rel}$ and $\text{weak-barbed-simulation} \text{ Rel Cal}$

shows $\text{weak-barbed-bisimulation} \text{ Rel Cal}$

If a relation as well as its inverse are weak simulations, then this relation is a weak bisimulation.
lemma weak-reduction-simulations-impl-bisimulation:
  fixes Rel :: (‘proc × ‘proc) set
  and Cal :: ‘proc processCalculus
  assumes sim: weak-reduction-simulation Rel Cal
  and simInv: weak-reduction-simulation (Rel⁻¹) Cal
  shows weak-reduction-bisimulation Rel Cal
(proof)

lemma weak-reduction-bisimulations-impl-inverse-is-simulation:
  fixes Rel :: (‘proc × ‘proc) set
  and Cal :: ‘proc processCalculus
  assumes bisim: weak-reduction-bisimulation Rel Cal
  shows weak-reduction-simulation (Rel⁻¹) Cal
(proof)

lemma weak-reduction-simulations-iff-bisimulation:
  fixes Rel :: (‘proc × ‘proc) set
  and Cal :: ‘proc processCalculus
  shows (weak-reduction-simulation Rel Cal \and\ weak-reduction-simulation (Rel⁻¹) Cal) = weak-reduction-bisimulation Rel Cal
(proof)

lemma weak-barbed-simulations-iff-bisimulation:
  fixes Rel :: (‘proc × ‘proc) set
  and CWB :: (‘proc, ‘barbs) calculusWithBarbs
  shows (weak-barbed-simulation Rel CWB \and\ weak-barbed-simulation (Rel⁻¹) CWB) = weak-barbed-bisimulation Rel CWB
(proof)

A weak bisimulation is a weak correspondence simulation.

lemma weak-reduction-bisimulation-is-correspondence-simulation:
  fixes Rel :: (‘proc × ‘proc) set
  and Cal :: ‘proc processCalculus
  assumes bisim: weak-reduction-bisimulation Rel Cal
  shows weak-reduction-correspondence-simulation Rel Cal
(proof)

lemma weak-barbed-bisimulation-is-correspondence-simulation:
  fixes Rel :: (‘proc × ‘proc) set
  and CWB :: (‘proc, ‘barbs) calculusWithBarbs
  assumes bisim: weak-barbed-bisimulation Rel CWB
  shows weak-barbed-correspondence-simulation Rel CWB
(proof)

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

lemma weak-reduction-bisimulation-and-closures:
  fixes Rel :: (‘proc × ‘proc) set
  and Cal :: ‘proc processCalculus
  assumes bisim: weak-reduction-bisimulation Rel Cal
  shows weak-reduction-bisimulation (Rel⁺) Cal
  and weak-reduction-bisimulation (symcl Rel) Cal
  and weak-reduction-bisimulation (Rel⁺) Cal
  and weak-reduction-bisimulation (symcl (Rel⁺)) Cal
  and weak-reduction-bisimulation (Rel⁺) Cal
  and weak-reduction-bisimulation ((symcl (Rel⁺))⁺) Cal
(proof)

lemma weak-barbed-bisimulation-and-closures:
  fixes Rel :: (‘proc × ‘proc) set
  and CWB :: (‘proc, ‘barbs) calculusWithBarbs
(proof)
assumes bisim: weak-barbed-bisimulation Rel CWB
shows weak-barbed-bisimulation (Rel\textsuperscript{−1}) CWB
and weak-barbed-bisimulation (symcl Rel) CWB
and weak-barbed-bisimulation (Rel\textsuperscript{+}) CWB
and weak-barbed-bisimulation (symcl (Rel\textsuperscript{−1})) CWB
and weak-barbed-bisimulation (Rel\textsuperscript{*}) CWB
and weak-barbed-bisimulation ((symcl (Rel\textsuperscript{−1}))\textsuperscript{+}) CWB
⟨proof⟩

A strong reduction bisimulation is relation $R$ such that
\begin{itemize}
  \item 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 $(P', Q') \in R$,
  \item 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 $(P', Q') \in R$.
\end{itemize}

abbreviation strong-reduction-bisimulation
:: ('proc × 'proc) set ⇒ bool
where
strong-reduction-bisimulation Rel Cal ≡
(∀ P Q P', (P, Q) ∈ Rel ∧ $P \longrightarrow Cal P'$ → (∃ Q', Q → Cal Q' ∧ (P', Q') ∈ Rel))
∧ (∀ P Q Q'. (P, Q) ∈ Rel ∧ $Q \longrightarrow Cal Q'$ → (∃ P', P → Cal P' ∧ (P', Q') ∈ Rel))

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

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

A symmetric strong simulation is a strong bisimulation.

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

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

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

lemma strong-reduction-simulations-impl-bisimulation:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes sim: strong-reduction-simulation Rel Cal
and simInv: strong-reduction-simulation (Rel\textsuperscript{−1}) Cal
shows strong-reduction-bisimulation Rel Cal
⟨proof⟩

lemma strong-reduction-bisimulations-impl-inverse-is-simulation:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes bisim: strong-reduction-bisimulation Rel Cal
shows strong-reduction-simulation (Rel\textsuperscript{−1}) Cal
⟨proof⟩
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 (symcl (Rel⁺)) Cal
and strong-reduction-bisimulation (symcl (Rel⁻¹)) Cal
and strong-reduction-bisimulation ((symcl (Rel⁻¹))⁺) 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 (symcl (Rel⁺)) CWB
and strong-barbed-bisimulation (symcl (Rel⁻¹)) CWB
and strong-barbed-bisimulation ((symcl (Rel⁻¹))⁺) 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.

\[
\text{inductive-set } \text{stepsClosure} :: \left('a \times 'a\right) \text{ set} \Rightarrow \text{'}a \text{ processCalculus} \Rightarrow \left('a \times 'a\right) \text{ set}
\]

\[
\begin{align*}
\text{for } & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and } & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{where} & \\
\text{rel: } & (P, Q) \in \text{Rel} \implies (P, Q) \in \text{stepsClosure} \text{ Rel Cal} \\
\text{steps: } & P \rightarrow \text{Cal} \rightarrow P' \implies (P', P) \in \text{stepsClosure} \text{ Rel Cal} \\
\text{trans: } & [(P, Q) \in \text{stepsClosure} \text{ Rel Cal}; (Q, R) \in \text{stepsClosure} \text{ Rel Cal}] \\
& \implies (P, R) \in \text{stepsClosure} \text{ Rel Cal}
\end{align*}
\]

\[
\text{abbreviation } \text{stepsClosureInfix} :: \\
\left('a \Rightarrow \left('a \times 'a\right) \text{ set} \Rightarrow 'a \text{ processCalculus} \Rightarrow 'a \Rightarrow \text{bool}\right)\left(R \mapsto \rightarrow <\cdot,\cdot>\right)\left[75, 75, 75, 75\right] 80
\]

\[
\begin{align*}
\text{where} & \\
P \mapsto <\text{Rel},\text{Cal}> Q \equiv (P, Q) \in \text{stepsClosure} \text{ Rel Cal}
\end{align*}
\]

Applying the steps closure twice does not change the relation.

\[
\text{lemma steps-closure-of-steps-closure:} \\
\begin{align*}
\text{fixes} & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and} & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{shows} & \text{stepsClosure} \left(\text{stepsClosure} \text{ Rel Cal}\right) \text{ Cal} = \text{stepsClosure} \text{ Rel Cal}
\end{align*}
\]

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

The steps closure is a preorder.

\[
\text{lemma stepsClosure-refl:} \\
\begin{align*}
\text{fixes} & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and} & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{shows} & \text{refl} \left(\text{stepsClosure} \text{ Rel Cal}\right)
\end{align*}
\]

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

\[
\text{lemma refl-trans-closure-of-rel-impl-steps-closure:} \\
\begin{align*}
\text{fixes} & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and} & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{and} & P Q :: 'a \\
\text{assumes} & (P, Q) \in \text{Rel}^* \\
\text{shows} & P \mapsto <\text{Rel},\text{Cal}> Q
\end{align*}
\]

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

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

\[
\text{lemma steps-closure-is-weak-reduction-simulation:} \\
\begin{align*}
\text{fixes} & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and} & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{shows} & \text{weak-reduction-simulation} \left(\text{stepsClosure} \text{ Rel Cal}\right) \text{ Cal}
\end{align*}
\]

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

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

\[
\text{lemma inverse-contrasimulation-impl-reverse-pair-in-steps-closure:} \\
\begin{align*}
\text{fixes} & \text{Rel} :: \left('a \times 'a\right) \text{ set} \\
\text{and} & \text{Cal} :: \text{'}a \text{ processCalculus} \\
\text{and} & P Q :: 'a \\
\text{assumes} & \text{con: weak-reduction-contrasimulation} \left(\text{Rel}^{-1}\right) \text{ Cal} \\
\text{and} & \text{pair: (P, Q) } \in \text{Rel} \\
\text{shows} & Q \mapsto <\text{Rel},\text{Cal}> P
\end{align*}
\]

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

\[
\text{lemma simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation:}
\]
Accordingly, if Rel is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of Rel is a coupled simulation.

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

- **Fixes** Rel :: ('a × 'a) set
- **And** Cal :: 'a processCalculus
- **Assumes** sim: weak-reduction-simulation Rel Cal and con: weak-reduction-contrasimulation (Rel⁻¹) Cal
- **Shows** weak-reduction-coupled-simulation (stepsClosure Rel Cal) Cal

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

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

- **Fixes** Rel :: ('a × 'a) set
- **And** Cal :: 'a processCalculus
- **And** P Q :: 'a
- **Assumes** A1: P R→→<Rel, Cal> Q and A2: weak-reduction-simulation Rel Cal
- **Shows** ∃Q'. Q →→Cal* Q' ∧ (P, Q') ∈ Rel*

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

- **Fixes** Rel :: ('a × 'a) set
- **And** Cal :: 'a processCalculus
- **And** P Q :: 'a
- **Assumes** A1: P R→→<Rel⁻¹, Cal> Q and A2: weak-reduction-contrasimulation (Rel⁻¹) Cal
- **Shows** ∃Q'. Q →→Cal* Q' ∧ (Q', P) ∈ Rel*

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

- **Fixes** Rel :: ('a × 'a) set
- **And** Cal :: 'a processCalculus
- **And** P Q :: 'a
- **Assumes** A1: P R→→<Rel⁻¹, Cal> Q and A2: weak-reduction-contrasimulation (Rel⁻¹) Cal
- **Shows** ∃Q'. Q →→Cal* Q' ∧ (P, Q') ∈ Rel*

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 ∋
(∃ 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 ↦−−ST (STCal Source Target) P' = P ↦−−ST P'
(proof)
lemma STStep-step:
  fixes S :: 'procS
  and T :: 'procT
  and P' :: ('procS, 'procT) Proc
  shows SourceTerm S \rightarrow ST \cdot P' = (\exists S'. S' \in S \cdot P' \wedge S \rightarrow Source S')
  and TargetTerm T \rightarrow ST \cdot P' = (\exists T'. T' \in T \cdot P' \wedge T \rightarrow Target T')
  (proof)

lemma STCal-step:
  fixes S :: 'procS
  and T :: 'procT
  and P' :: ('procS, 'procT) Proc
  shows SourceTerm S \rightarrow (STCal Source Target) \cdot P' = (\exists S'. S' \in S \cdot P' \wedge S \rightarrow Source S')
  and TargetTerm T \rightarrow (STCal Source Target) \cdot P' = (\exists T'. T' \in T \cdot P' \wedge T \rightarrow Target T')
  (proof)

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

abbreviation stepsST
  :: ('procS, 'procT) Proc \Rightarrow ('procS, 'procT) Proc \Rightarrow bool \ (\isasymcdot \isasymrightarrow ST^* \cdot [70, 70] 80)
  where
  \isasymforall P \isasymrightarrow ST^* \cdot P' \equiv
  (\exists S'. S \in S \cdot P' \wedge S \rightarrow Source S') \vee (\exists T', T \in T \cdot P' \wedge T \rightarrow Target T')

lemma STSteps-steps:
  fixes S :: 'procS
  and T :: 'procT
  and P' :: ('procS, 'procT) Proc
  shows SourceTerm S \rightarrow ST^* \cdot P' = (\exists S'. S' \in S \cdot P' \wedge S \rightarrow Source^* S')
  and TargetTerm T \rightarrow ST^* \cdot P' = (\exists T'. T' \in T \cdot P' \wedge T \rightarrow Target^* T')
  (proof)

lemma STCal-steps:
  fixes S :: 'procS
  and T :: 'procT
  and P' :: ('procS, 'procT) Proc
  shows SourceTerm S \rightarrow (STCal Source Target)^* \cdot P' = (\exists S'. S' \in S \cdot P' \wedge S \rightarrow Source^* S')
  and TargetTerm T \rightarrow (STCal Source Target)^* \cdot P' = (\exists T'. T' \in T \cdot P' \wedge T \rightarrow Target^* T')
  (proof)

lemma stepsST-STCal-steps:
  fixes P P' :: ('procS, 'procT) Proc
  shows P \rightarrow (STCal Source Target)^* \cdot P' = P \rightarrow ST^* \cdot P'
  (proof)

lemma stepsST-refl:
  fixes P :: ('procS, 'procT) Proc
  shows P \rightarrow ST^* \cdot P
  (proof)

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

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

abbreviation divergentST
\[ (\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool} (\xrightarrow{-\rightarrow\ ST \omega} [70] 80) \]

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

**lemma** \( \text{STCal-divergent} \):

fixes \( S \:: \text{'procS} \)
and \( T :: \text{'procT} \)

shows \( \text{SourceTerm} S \xrightarrow{\text{STCal Source Target} \omega} = S \xrightarrow{\text{Source} \omega} \)
and \( \text{TargetTerm} T \xrightarrow{\text{STCal Source Target} \omega} = T \xrightarrow{\text{Target} \omega} \)

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

**lemma** \( \text{divergentST-STCal-divergent} \):

fixes \( P :: (\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool} \)

shows \( P \xrightarrow{\text{STCal Source Target} \omega} = P \xrightarrow{\text{ST} \omega} \)

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

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

\textbf{abbreviation} \( \text{enc-preserves-pred} :: ((\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool}) \Rightarrow \text{bool} \)

where
\[ \text{enc-preserves-pred} \text{ Pred} \equiv \forall S. \text{Pred} (\text{SourceTerm} S) \xrightarrow{-\rightarrow\ ST \omega} \text{Pred} (\text{TargetTerm} ([S])) \]

\textbf{abbreviation} \( \text{enc-preserves-binary-pred} :: ((\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool}) \Rightarrow \text{bool} \)

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

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

\textbf{abbreviation} \( \text{enc-reflects-pred} :: ((\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool}) \Rightarrow \text{bool} \)

where
\[ \text{enc-reflects-pred} \text{ Pred} \equiv \forall S. \text{Pred} (\text{TargetTerm} ([S])) x \xrightarrow{-\rightarrow\ ST \omega} \text{Pred} (\text{SourceTerm} S) x \]

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

\textbf{abbreviation} \( \text{enc-respects-pred} :: ((\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool}) \Rightarrow \text{bool} \)

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

\textbf{abbreviation} \( \text{enc-respects-binary-pred} :: ((\text{'procS}, \text{'procT}) \xrightarrow{\text{Proc}} \text{bool}) \Rightarrow \text{bool} \)

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

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.

\textbf{locale} \( \text{encoding-wrt-barbs} = \)

\textbf{encoding} \( \text{Source} :: \text{'procS} \xrightarrow{\text{processCalculus}} \)
and \( \text{Target} :: \text{'procT} \xrightarrow{\text{processCalculus}} \)
and \( \text{Enc} :: \text{'procS} \Rightarrow \text{'procT} + \)

fixes \( \text{SWB} :: (\text{'procS}, \text{'barbs}) \xrightarrow{\text{calculusWithBarbs}} \)
and \( \text{TWB} :: (\text{'procT}, \text{'barbs}) \xrightarrow{\text{calculusWithBarbs}} \)

\textbf{assumes} \( \text{calS}: \text{calculusWithBarbs. Calculus SWB} = \text{Source} \)
and \( \text{calT}: \text{calculusWithBarbs. Calculus TWB} = \text{Target} \)

\textbf{begin}

24
lemma STCalWB-STCal:
  shows Calculus (STCalWB SWB TWB) = STCal Source Target
  ⟨proof⟩

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

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

lemma STCalWB-hasBarbST:
  fixes P :: ('procS, 'procT) Proc
  and a :: 'barbs
  shows P↓<STCalWB SWB TWB>a = P↓.a
  ⟨proof⟩

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

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

lemma respection-of-barbs-in-barbed-encoding:
  fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  and P Q :: ('procS, 'procT) Proc
  and a :: 'barbs
  assumes respection: rel-respects-barbs Rel (STCalWB SWB TWB)
  and rel: (P, Q) ∈ Rel
  shows P↓.a = Q↓.a
  ⟨proof⟩

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

abbreviation reachesBarbST
:: ('procS, 'procT) Proc ⇒ 'barbs ⇒ bool
where
P↓.a ≡ (∃ S. S ∈ S P ∧ S↓<SWB>a) ∨ (∃ T. T ∈ T P ∧ T↓<TWB>a)

lemma STCalWB-reachesBarbST:
  fixes P :: ('procS, 'procT) Proc
  and a :: 'barbs
  shows P↓<STCalWB SWB TWB>a = P↓.a
  ⟨proof⟩
Lemma weak-preservation-of-barbs-in-barbed-encoding:

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and P Q :: ('procS, 'procT) Proc
and a :: 'barbs

assumes preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)
and rel: (P, Q) ∈ Rel
and barb: P浮现 a

shows Q浮现 a

(proof)

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

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and P Q :: ('procS, 'procT) Proc
and a :: 'barbs

assumes reflection: rel-weakly-reflects-barbs Rel (STCalWB SWB TWB)
and rel: (P, Q) ∈ Rel
and barb: Q浮现 a

shows P浮现 a

(proof)

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

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and P Q :: ('procS, 'procT) Proc
and a :: 'barbs

assumes respection: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
and rel: (P, Q) ∈ Rel

shows P浮现 a = Q浮现 a

(proof)

End

Theory SourceTargetRelation

Imports Encodings SimulationRelations

Begin

5 Relation between Source and Target Terms

5.1 Relations Induced by the Encoding Function

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

Inductive-set (in encoding) indRelR
where
encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelR

Abbreviation (in encoding) indRelRinfix ::
where
P R[.]R Q ⇒ (P, Q) ∈ indRelR

Inductive-set (in encoding) indRelRPO
where
encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelRPO |
source: (SourceTerm S, SourceTerm S) ∈ indRelRPO |
target: \((\text{TargetTerm } T, \text{TargetTerm } T) \in \text{indRelRPO} \mid\)
trans: \([([P, Q) \in \text{indRelRPO}; (Q, R) \in \text{indRelRPO}] \implies (P, R) \in \text{indRelRPO})\)

**abbreviation (in encoding)** \(\text{indRelRPOinf}::\)
\((\text{procS}, \text{procT}) \text{ Proc} \Rightarrow (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow \text{bool} (-\subseteq [75, 75] 80)\)
where \(P \subseteq [\text{R} Q \equiv (P, Q) \in \text{indRelRPO}\)

**lemma (in encoding)** \(\text{indRelRPO-refl}:\)
shows refl \(\text{indRelRPO}\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{indRelRPO-is-preorder}:\)
shows preorder \(\text{indRelRPO}\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{refl-trans-closure-of-indRelR}:\)
shows \(\text{indRelRPO} = \text{indRelR}^*\)
\(\langle \text{proof} \rangle\)

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

**lemma (in encoding)** \(\text{indRelR-impl-exists-source-target-relation}:\)
fixes \(\text{PredA} \:: \((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \text{ set} \Rightarrow \text{bool}\)
and \(\text{PredB} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \Rightarrow \text{bool}\)
shows \(\text{PredA} \text{ indRelR} \Rightarrow \exists \text{ Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land \text{PredA} \text{ Rel}\)
and \(\forall (P, Q) \in \text{indRelR}. \text{PredB} (P, Q)\)
\(\implies \exists \text{ Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{PredB} (P, Q))\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{source-target-relation-impl-indRelR}:\)
fixes \(\text{Rel} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \text{ set}\)
and \(\text{Pred} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \Rightarrow \text{bool}\)
assumes \(\text{encRel}: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
and \(\text{condRel}: \forall (P, Q) \in \text{Rel}. \text{Pred} (P, Q)\)
shows \(\forall (P, Q) \in \text{indRelR}. \text{Pred} (P, Q)\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{indRelR-iff-exists-source-target-relation}:\)
fixes \(\text{Pred} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \Rightarrow \text{bool}\)
shows \(\forall (P, Q) \in \text{indRelR}. \text{Pred} (P, Q)\)
\(\equiv (\exists \text{ Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{Pred} (P, Q)))\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{indRelR-modulo-pred-impl-indRelRPO-modulo-pred}:\)
fixes \(\text{Pred} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \Rightarrow \text{bool}\)
assumes \(\text{reflCond}: \forall P. \text{Pred} (P, P)\)
and \(\text{transCond}: \forall P Q R. \text{Pred} (P, Q) \land \text{Pred} (Q, R) \implies \text{Pred} (P, R)\)
shows \(\forall (P, Q) \in \text{indRelR}. \text{Pred} (P, Q) = (\forall (P, Q) \in \text{indRelRPO}. \text{Pred} (P, Q))\)
\(\langle \text{proof} \rangle\)

**lemma (in encoding)** \(\text{indRelRPO-iff-exists-source-target-relation}:\)
fixes \(\text{Pred} :: ((\text{procS}, \text{procT}) \text{ Proc} \times (\text{procS}, \text{procT}) \text{ Proc}) \Rightarrow \text{bool}\)
shows \(\forall (P, Q) \in \text{indRelRPO}. \text{Pred} (P, Q) = (\exists \text{ Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{Pred} (P, Q)) \land \text{preorder Rel})\)
\(\langle \text{proof} \rangle\)

An encoding preserves, reflects, or respects a predicate iff \(\text{indRelR}\) preserves, reflects, or respects this predicate.
lemma (in encoding) enc-satisfies-pred-impl-indRelR-satisfies-pred:
  fixes Pred :: ('procS, 'procT) Proc × ('procS, 'procT) Proc ⇒ bool
  assumes encCond: ∀ S. Pred (SourceTerm S, TargetTerm ([S]))
  shows ∀ (P, Q) ∈ indRelR. Pred (P, Q)
    ⟨proof⟩

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

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

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

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

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

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

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

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

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

lemma (in encoding) enc-respects-binary-pred-iff-indRelR-respects-binary-pred:
  fixes Pred :: ('procS, 'procT) Proc ⇒ bool
  shows enc-respects-binary-pred Pred = rel-respects-binary-pred indRelR Pred
    ⟨proof⟩
To analyse the reflection of source term behaviours we use relations that contain the pairs \((\text{enc } S, S)\) for all source terms \(S\).
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 ::
where
P ≲[·]L Q ≡ (P, Q) ∈ indRelLPO

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

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

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

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

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

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

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

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

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

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

lemma (in encoding) indRelR-respects-iff-indRelL-respects:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows rel-respects indRelR Pred = rel-respects indRelL Pred
⟨proof⟩
shows \( \text{rel-respects-pred indRelR Pred} = \text{rel-respects-pred indRelL Pred} \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( \text{rel-respects-binary-pred indRelR Pred} = \text{rel-respects-binary-pred indRelL Pred} \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-preserves-pred Rel Pred}) \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-preserves-binary-pred Rel Pred}) \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-preserves-pred Rel Pred}) \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-preserves-binary-pred Rel Pred}) \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-respects-pred Rel Pred}) \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{’b} \Rightarrow \text{bool} \)
- **Shows** \( (\exists \text{ Rel}. (\forall \text{ S. (SourceTerm S, TargetTerm ([S]])} \in \text{ Rel}) \land \text{rel-respects-binary-pred Rel Pred}) \)

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

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{bool} \)
- **Shows** \( \text{enc-preserves-pred Pred} = \text{rel-reflects-pred indRelL Pred} \)

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

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

- **Fixes** \( \text{Pred} :: (\text{’procS}, \text{’procT}) \text{ Proc} \Rightarrow \text{bool} \)
- **Shows** \( \text{enc-reflects-pred Pred} = \text{rel-preserves-pred indRelL Pred} \)

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

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

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

- **Fixes**: $\text{Pred} :: (\langle \text{procS}, \text{procT} \rangle \text{ Proc} \Rightarrow \text{bool})$
- **Shows**: $\text{enc-preserves-pred Pred} = \text{rel-preserves-pred indRelL Pred}$

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

- **Inductive-set (in encoding)** $\text{indRel}$
  
  :: $(\langle \langle \text{procS}, \text{procT} \rangle \text{ Proc} \rangle \times (\langle \text{procS}, \text{procT} \rangle \text{ Proc}))$ set

- **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)** $\text{indRelInfix}$ :

  $(\langle \text{procS}, \text{procT} \rangle \text{ Proc} \Rightarrow (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow \text{bool} (- R[.] - [75, 75] 80))$

- **Where**
  
  $P R[.] Q \equiv (P, Q) \in \text{indRel}$

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

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

  (proof)

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

- **Inductive-set (in encoding)** $\text{indRelEQ}$
  
  :: $(\langle \langle \text{procS}, \text{procT} \rangle \text{ Proc} \rangle \times (\langle \text{procS}, \text{procT} \rangle \text{ Proc}))$ set

- **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)** $\text{indRelEQInfix}$ :

  $(\langle \text{procS}, \text{procT} \rangle \text{ Proc} \Rightarrow (\text{procS}, \text{procT}) \text{ Proc} \Rightarrow \text{bool} (- \sim[.] - [75, 75] 80))$

- **Where**
  
  $P \sim[.] Q \equiv (P, Q) \in \text{indRelEQ}$

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

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

  (proof)

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

- **Shows**: $\text{preorder indRelEQ}$
lemma (in encoding) indRelEQ-symm:
  shows $\text{symm indRelEQ}$
  (proof)

lemma (in encoding) indRelEQ-is-equivalence:
  shows $\text{equivalence indRelEQ}$
  (proof)

lemma (in encoding) refl-trans-closure-of-indRel:
  shows $\text{indRelEQ} = \text{indRel}^*$
  (proof)

lemma (in encoding) refl-symm-trans-closure-of-indRel:
  shows $\text{indRelEQ} = (\text{symcl (indRelR)})^+$
  (proof)

lemma (in encoding) symm-closure-of-indRelR:
  shows $\text{indRel} = \text{symcl indRelR}$
  and $\text{indRelEQ} = (\text{symcl (indRel^R)})^+$
  (proof)

lemma (in encoding) symm-closure-of-indRelL:
  shows $\text{indRel} = \text{symcl indRelL}$
  and $\text{indRelEQ} = (\text{symcl (indRelL^R)})^+$
  (proof)

The relation $\text{indRel}$ is a combination of $\text{indRelL}$ and $\text{indRelR}$. $\text{indRel}$ respects a predicate iff $\text{indRelR}$
  (or $\text{indRelL}$) respects it.

lemma (in encoding) indRel-respects-pred-iff-indRelR-respects-pred:
  fixes $\text{Pred} :: \left( \text{procS}, \text{procT} \right) \text{ Proc} \Rightarrow \text{bool}$
  shows $\text{rel-respects-pred indRel Pred = rel-respects-pred indRelR Pred}$
  (proof)

lemma (in encoding) indRel-respects-binary-pred-iff-indRelR-respects-binary-pred:
  fixes $\text{Pred} :: \left( \text{procS}, \text{procT} \right) \text{ Proc} \Rightarrow \text{b} \Rightarrow \text{bool}$
  shows $\text{rel-respects-binary-pred indRel Pred = rel-respects-binary-pred indRelR Pred}$
  (proof)

lemma (in encoding) indRel-cond-respection-iff-indRelR-cond-respection:
  fixes $\text{Pred} :: \left( \text{procS}, \text{procT} \right) \text{ Proc} \Rightarrow \text{bool}$
  shows $(\exists \text{Rel}).
\left( \forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel} \land (\text{TargetTerm ([S]], SourceTerm S) \in \text{Rel})
\land \text{rel-respects-pred Rel Pred})
\right) = (\exists \text{Rel}. (\forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel} \land \text{rel-respects-pred Rel Pred})
\land \text{rel-respects-binary-pred Rel Pred})$
  (proof)

lemma (in encoding) indRel-cond-binary-respection-iff-indRelR-cond-binary-respection:
  fixes $\text{Pred} :: \left( \text{procS}, \text{procT} \right) \text{ Proc} \Rightarrow \text{b} \Rightarrow \text{bool}$
  shows $(\exists \text{Rel}.
\left( \forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel} \land (\text{TargetTerm ([S]], SourceTerm S) \in \text{Rel})
\land \text{rel-respects-binary-pred Rel Pred})
\right) = (\exists \text{Rel}. (\forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel})
\land \text{rel-respects-binary-pred Rel Pred})$
  (proof)

An encoding respects a predicate iff $\text{indRel}$ respects this predicate.

lemma (in encoding) enc-respects-pred-iff-indRel-respects-pred:
  fixes $\text{Pred} :: \left( \text{procS}, \text{procT} \right) \text{ Proc} \Rightarrow \text{bool}$
  shows $(\exists \text{Rel}).
\left( \forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel} \land (\text{TargetTerm ([S]], SourceTerm S) \in \text{Rel})
\land \text{rel-respects-binary-pred Rel Pred})
\right) = (\exists \text{Rel}. (\forall S. (\text{SourceTerm S, TargetTerm ([S]]) \in \text{Rel})
\land \text{rel-respects-binary-pred Rel Pred})$
  (proof)

33
shows \( \text{enc-respects-pred Pred = rel-respects-pred indRel Pred} \)

(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) enc-respects-pred-iff-source-target-rel-respects-pred-encRL:**

- **fixes** \( \text{Pred :: ('procS, 'procT) Proc ⇒ bool} \)
- **shows** \( \text{enc-respects-pred Pred} \)
  \[ \begin{align*}
  &= (∃ \text{Rel}.
  \quad (∀ \text{S}. (\text{SourceTerm S}, \text{TargetTerm } ([\text{S}])) ∈ \text{Rel} ∧ (\text{TargetTerm } ([\text{S}]), \text{SourceTerm S}) ∈ \text{Rel})
  ∧ \text{rel-respects-pred Rel Pred})
  \end{align*} \)

(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 disjunct union of source and target terms.

**inductive-set (in encoding) indRelRT:**

\[ \begin{align*}
  &\text{ :: ('procT × 'procT) set ⇒ } ((\text{('procS, 'procT) Proc}) × ((\text{('procS, 'procT) Proc}) set
  
  \text{for TRel :: ('procT × 'procT) set}
  
  \text{where}
  
  \text{encR: } (\text{SourceTerm S}, \text{TargetTerm } ([\text{S}])) ∈ \text{indRelRT TRel} \mid
  
  \text{target: } (\text{T1, T2}) ∈ \text{TRel} \implies (\text{TargetTerm T1, TargetTerm T2}) ∈ \text{indRelRT TRel}
  \end{align*} \]

**abbreviation (in encoding) indRelRTinfix:**

\[ \begin{align*}
  &\text{ :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool}
  
  (- \text{ R[.]RT<-> - [75, 75, 75] 80})
  
  \text{where}
  
  P \text{ R[.]RT<TRel> Q ≡ (P, Q) ∈ indRelRT TRel}
  \end{align*} \]

**inductive-set (in encoding) indRelRTPO:**

\[ \begin{align*}
  &\text{ :: ('procT × 'procT) set ⇒ } ((\text{('procS, 'procT) Proc}) × ((\text{('procS, 'procT) Proc}) set
  
  \text{for TRel :: ('procT × 'procT) set}
  
  \text{where}
  
  \text{encR: } (\text{SourceTerm S}, \text{TargetTerm } ([\text{S}])) ∈ \text{indRelRTPO TRel} \mid
  
  \text{source: } (\text{SourceTerm S}, \text{SourceTerm S}) ∈ \text{indRelRTPO TRel} \mid
  
  \text{target: } (\text{T1, T2}) ∈ \text{TRel} \implies (\text{TargetTerm T1, TargetTerm T2}) ∈ \text{indRelRTPO TRel} \mid
  
  \text{trans: } ([\text{P, Q}] \text{ ∈ indRelRTPO TRel}) \implies (\text{Q, R}) ∈ \text{indRelRTPO TRel}
  \end{align*} \]

**abbreviation (in encoding) indRelRTPOinfix:**

\[ \begin{align*}
  &\text{ :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool}
  
  (- \text{ Z[.]RT<-> - [75, 75, 75] 80})
  
  \text{where}
  
  P \text{ Z[.]RT<TRel> Q ≡ (P, Q) ∈ indRelRTPO TRel}
  \end{align*} \]

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

- **fixes** \( \text{TRel :: ('procT × 'procT) set} \)
- **assumes** \( \text{refl TRel} \)
- **shows** \( \text{refl (indRelRTPO TRel)} \)

(proof)

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

- **fixes** \( \text{TRel :: ('procT × 'procT) set} \)
- **assumes** \( \text{refl TRel} \)
- **shows** \( \text{indRelRTPO TRel = (indRelRT TRel)} \)

(proof)

**lemma (in encoding) indRelRTPO-is-preorder:**
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) transitive-closure-of-\( \text{TRel} \)-to-\( \text{indRelRT} \):**

- **Fixes** \( \text{TRel} := \langle \text{procT} \times \text{procT} \rangle \) set
- **Assumes** \( \text{reflT} : \text{refl} \text{TRel} \)
- **Shows** \( \text{preorder} (\text{indRelRT} \text{PO} \text{TRel}) \)

\[
\text{(proof)}
\]

**Lemma (in encoding) \( \text{indRelRT} \)-iff-exists-source-target-relation:**

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

\[
\text{(proof)}
\]

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

- **Fixes** \( \text{TRel} := \langle \text{procT} \times \text{procT} \rangle \) set
- **Assumes** \( \text{reflCond} : \forall \text{P. Pred (P, P)} \)
- **Shows** \( \langle (\forall \text{P Q R. Pred (P, Q) } \&\& \text{Pred (Q, R)} \Rightarrow \text{Pred (P, R)}) \rangle \)

\[
\text{(proof)}
\]

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

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

- \( \langle \text{procT} \times \text{procT} \rangle \) set
- \( \forall \text{TRel := \langle \text{procT} \times \text{procT} \rangle \) set
- \( \text{encL} : \langle \text{TargetTerm} ([S]), \text{SourceTerm} S \rangle \in \text{indRelLT} \text{TRel} | \)
- \( \text{target}: (T1, T2) \in \text{TRel} \implies (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{indRelLT} \text{TRel} \)

**Abbreviation (in encoding) \( \text{indRelLTinf} \):**

- \( \langle \text{procS}, \text{procT} \rangle \text{Proc} \Rightarrow \langle \text{procT} \times \text{procT} \rangle \) set \( \Rightarrow \langle \text{procS}, \text{procT} \rangle \text{Proc} \Rightarrow \text{bool} \)
- \( \text{- R[\text{LT}<\text{- [75, 75, 75] 80}]} \)
- \( \text{- P R[\text{LT}<\text{TRel> Q} \equiv (P, Q) \in \text{indRelLT} \text{TRel}} \)

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

- \( \langle \text{procT} \times \text{procT} \rangle \) set \( \Rightarrow \langle \langle \text{procS}, \text{procT} \rangle \text{Proc} \times \langle \text{procS}, \text{procT} \rangle \text{Proc} \rangle \) set
for \( TRel :: (\textquote{procT} \times \textquote{procT}) \text{ set} \)

where

\[
\text{encl: } (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{indRelLTPO} \ TRel \ |
\text{source: } (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{indRelLTPO} \ TRel \ |
\text{target: } (T1, T2) \in TRel \quad \Rightarrow \quad (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{indRelLTPO} \ TRel \ |\n\text{trans: } [[(P, Q)] \in \text{indRelLTPO} \ TRel; (Q, R) \in \text{indRelLTPO} \ TRel] \quad \Rightarrow \quad (P, R) \in \text{indRelLTPO} \ TRel
\]

abbreviation \text{(in encoding)} \text{indRelLTPOinfix}

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

\[
\begin{align*}
& (- \bowtie LTL \text{\textquote{\textless}}} - [75, 75, 75] 80) \\
\text{where} \\
& P \bowtie LTL \text{\textquote{\textless}} Q \equiv (P, Q) \in \text{indRelLTPO} \ TRel
\end{align*}
\]

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

\[
\begin{align*}
& \text{fixes } TRel :: (\textquote{procT} \times \textquote{procT}) \text{ set} \\
& \text{assumes } \text{refl: } \text{refl} \ TRel \\
& \text{shows } \text{refl} (\text{indRelLTPO} \ TRel)
\end{align*}
\]

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

\[
\begin{align*}
& \text{fixes } TRel :: (\textquote{procT} \times \textquote{procT}) \text{ set} \\
& \text{assumes } \text{refl: } \text{refl} \ TRel \\
& \text{shows } \text{indRelLTPO} \ TRel = (\text{indRelLT} \ TRel)^\ast
\end{align*}
\]

(proof)

inductive-set \text{(in encoding)} \text{indRelT}

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

\[
\begin{align*}
& \text{for } TRel :: (\textquote{procT} \times \textquote{procT}) \text{ set} \\
& \text{where} \\
& \text{enclR: } (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{indRelT} \ TRel \\
& \text{enclL: } (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{indRelT} \ TRel \\
& \text{target: } (T1, T2) \in TRel \quad \Rightarrow \quad (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{indRelT} \ TRel
\end{align*}
\]

abbreviation \text{(in encoding)} \text{indRelTinfix}

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

\[
\begin{align*}
& (- \bowtie \text{T} \text{\textquote{\textless}} - [75, 75, 75] 80) \\
\text{where} \\
& P \bowtie \text{T} \text{\textquote{\textless}} Q \equiv (P, Q) \in \text{indRelT} \ TRel
\end{align*}
\]

lemma \text{(in encoding)} \text{indRelT-symm:}

\[
\begin{align*}
& \text{fixes } TRel :: (\textquote{procT} \times \textquote{procT}) \text{ set} \\
& \text{assumes } \text{symm: } \text{sym} \ TRel \\
& \text{shows } \text{sym} (\text{indRelT} \ TRel)
\end{align*}
\]

(proof)

inductive-set \text{(in encoding)} \text{indRelTEQ}

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

\[
\begin{align*}
& (- \bowtie \text{T} \text{\textquote{\textless}} - [75, 75, 75] 80) \\
\text{where} \\
& \text{enclR: } (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{indRelTEQ} \ TRel \\
& \text{enclL: } (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{indRelTEQ} \ TRel \\
& \text{target: } (T1, T2) \in TRel \quad \Rightarrow \quad (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{indRelTEQ} \ TRel \\
& \text{trans: } [[(P, Q)] \in \text{indRelTEQ} \ TRel; (Q, R) \in \text{indRelTEQ} \ TRel] \quad \Rightarrow \quad (P, R) \in \text{indRelTEQ} \ TRel
\end{align*}
\]

abbreviation \text{(in encoding)} \text{indRelTEQinfix}

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

\[
\begin{align*}
& (- \bowtie \text{\textless} - [75, 75, 75] 80) \\
\text{where} \\
& P \bowtie \text{\textless} Q \equiv (P, Q) \in \text{indRelTEQ} \ TRel
\end{align*}
\]

lemma \text{(in encoding)} \text{indRelTEQ-refl:}
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes refl: refl $T\text{Rel}$
shows refl $(\text{indRelTEQ} T\text{Rel})$

lemma (in encoding) $\text{indRelTEQ-symm}$:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes symm: sym $T\text{Rel}$
shows sym $(\text{indRelTEQ} T\text{Rel})$

lemma (in encoding) refl-trans-closure-of-indRelT:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes refl: refl $T\text{Rel}$
shows indRelTEQ $T\text{Rel} = (\text{indRelT} T\text{Rel})^*$

lemma (in encoding) refl-symm-trans-closure-of-indRelT:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes refl: refl $T\text{Rel}$
and symm: sym $T\text{Rel}$
shows indRelTEQ $T\text{Rel} = (\text{symcl} ((\text{indRelT} T\text{Rel})^*))^*$

lemma (in encoding) symm-closure-of-indRelRT:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes refl: refl $T\text{Rel}$
and symm: sym $T\text{Rel}$
shows indRelT $T\text{Rel} = \text{symcl} (\text{indRelRT} T\text{Rel})$
and indRelTEQ $T\text{Rel} = (\text{symcl} ((\text{indRelRT} T\text{Rel})^*))^*$

lemma (in encoding) symm-closure-of-indRelLT:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
assumes refl: refl $T\text{Rel}$
and symm: sym $T\text{Rel}$
shows indRelT $T\text{Rel} = \text{symcl} (\text{indRelLT} T\text{Rel})$
and indRelTEQ $T\text{Rel} = (\text{symcl} ((\text{indRelLT} T\text{Rel})^*))^*$

If the relations $\text{indRelRT}$, $\text{indRelLT}$, or $\text{indRelT}$ contain a pair of target terms, then this pair is also related by the considered target term relation.

lemma (in encoding) $\text{indRelRT-to-TRel}$:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
and $TP \ TQ : \text{\textquotesingle proc}\text{T}$
assumes rel: TargetTerm $TP \text{ Rel} [\cdot \cdot RT < T\text{Rel}]$ TargetTerm $TQ$
shows $(TP, TQ) \in T\text{Rel}$

lemma (in encoding) $\text{indRelLT-to-TRel}$:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
and $TP \ TQ : \text{\textquotesingle proc}\text{T}$
assumes rel: TargetTerm $TP \text{ Rel} [\cdot \cdot LT < T\text{Rel}]$ TargetTerm $TQ$
shows $(TP, TQ) \in T\text{Rel}$

lemma (in encoding) $\text{indRelT-to-TRel}$:
fixes $T\text{Rel} : (\text{\textquotesingle proc}\text{T} \times \text{\textquotesingle proc}\text{T}) \text{ set}$
and $TP \ TQ : \text{\textquotesingle proc}\text{T}$
assumes rel: TargetTerm $TP \text{ Rel} [\cdot \cdot T < T\text{Rel}]$ TargetTerm $TQ$
If the preorders indRelRTPO, indRelLTPO, or the equivalence indRelTEQ contain a pair of terms, then the pair of target terms that is related to these two terms is also related by the reflexive and transitive closure of the considered target term relation.

**Lemma (in encoding) indRelRTPO-to-TRel:**
- **fixes** \( TRel : (\text{\textquotesingle}procT \times \text{\textquotesingle}procT) \text{ set} \)
- **and** \( P \ Q : (\text{\textquotesingle}procS, \text{\textquotesingle}procT) \text{ Proc} \)
- **assumes** rel: \( P \subseteq RT < TRel > Q \)
- **shows** \( \forall SP SQ. SP \in S P \wedge SQ \in S Q \implies SP = SQ \)
- **proof**

**Lemma (in encoding) indRelLTPO-to-TRel:**
- **fixes** \( TRel : (\text{\textquotesingle}procT \times \text{\textquotesingle}procT) \text{ set} \)
- **and** \( P \ Q : (\text{\textquotesingle}procS, \text{\textquotesingle}procT) \text{ Proc} \)
- **assumes** rel: \( P \subseteq LT < TRel > Q \)
- **shows** \( \forall SP SQ. SP \in S P \wedge SQ \in S Q \implies SP = SQ \)
- **proof**

**Lemma (in encoding) indRelTEQ-to-TRel:**
- **fixes** \( TRel : (\text{\textquotesingle}procT \times \text{\textquotesingle}procT) \text{ set} \)
- **and** \( P \ Q : (\text{\textquotesingle}procS, \text{\textquotesingle}procT) \text{ Proc} \)
- **assumes** rel: \( P \sim T < TRel > Q \)
- **shows** \( \forall SP SQ. SP \in S P \wedge SQ \in S Q \implies SP = SQ \)
- **proof**

**Lemma (in encoding) trans-closure-of-TRel-refl-cond:**
- **fixes** \( TRel : (\text{\textquotesingle}procT \times \text{\textquotesingle}procT) \text{ set} \)
- **and** \( TP \ TJ : (\text{\textquotesingle}procT \text{ Proc}) \)
- **assumes** \( (TP, TJ) \in (TRel \cup \{(T1, T2) \), \exists S. T1 = [S] \wedge T2 = [S])^+ \)
- **shows** \( (TP, TJ) \in TRel^+ \)
- **proof**

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

**Lemma (in encoding) indRelRTPO-relates-source-target:**
- **fixes** \( TRel : (\text{\textquotesingle}procT \times \text{\textquotesingle}procT) \text{ set} \)
- **and** \( S : \text{\textquotesingle}procS \text{ and} T : \text{\textquotesingle}procT \)
- **assumes** \( (TP, TJ) \in (TRel \cup \{(T1, T2) \), \exists S. T1 = [S] \wedge T2 = [S])^+ \)
- **shows** \( \text{\textquotesingle}TRel \text{\textquotesingle} \text{\textquotesingle}(\text{\textquotesingle}S\text{\textquotesingle}), \text{\textquotesingle}TargetTerm\text{\textquotesingle} \text{\textquotesingle}T\text{\textquotesingle}) \in (\text{\textquotesingle}indRelRTPO\text{\textquotesingle} TRel)^+ \)
- **proof**

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

**Lemma (in encoding-wrt-barbs)** rel-with-target-impl-TRel-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ set
- and $Rel : ('procS, 'procT) Proc \times ('procS, 'procT) Proc$ set
- assumes preservation: rel-preserves-barbs $Rel (STCalWB SWB TWB) \Rightarrow$ (TargetTerm $T1, TargetTerm T2) \in Rel$
- shows rel-preserves-barbs $TRel TWB$

(proof)

**Lemma (in encoding-wrt-barbs)** indRelRTPO-impl-TRel-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ set
- assumes preservation: rel-preserves-barbs $(indRelRTPO TRel) (STCalWB SWB TWB)$
- shows rel-preserves-barbs $TRel TWB$

(proof)

**Lemma (in encoding-wrt-barbs)** indRelLTPO-impl-TRel-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ set
- assumes preservation: rel-preserves-barbs $(indRelLTPO TRel) (STCalWB SWB TWB)$
- shows rel-preserves-barbs $TRel TWB$

(proof)

**Lemma (in encoding-wrt-barbs)** indRelTEQ-impl-TRel-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ set
- assumes preservation: rel-preserves-barbs $(indRelTEQ TRel) (STCalWB SWB TWB)$
- shows rel-preserves-barbs $TRel TWB$

(proof)

**Lemma (in encoding-wrt-barbs)** rel-with-target-impl-TRel-weakly-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ set
- and $Rel : ('procS, 'procT) Proc \times ('procS, 'procT) Proc$ set
- assumes preservation: rel-weakly-preserves-barbs $Rel (STCalWB SWB TWB) \Rightarrow$ (TargetTerm $T1, TargetTerm T2) \in Rel$
- shows rel-weakly-preserves-barbs $TRel TWB$

(proof)

**Lemma (in encoding-wrt-barbs)** indRelRTPO-impl-TRel-weakly-preserves-barbs:
- fixes $TRel : ('procT \times 'procT)$ 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 $TRel : ('procT \times 'procT)$ 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 $TRel : ('procT \times 'procT)$ set
- assumes preservation: rel-weakly-preserves-barbs $(indRelTEQ TRel) (STCalWB SWB TWB)$
- shows rel-weakly-preserves-barbs $TRel TWB$

(proof)

If indRelRTPO, indRelLTPO, or indRelTPO reflects barbs then so does the corresponding target term relation.
shows \( \text{rel-reflects-barbs} \) \( \text{TRel} \) \( \text{TWB} \)

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelRTPO-impl-TRel-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-reflects-barbs} \( (\text{indRelRTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelLTPO-impl-TRel-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-reflects-barbs} \( (\text{indRelLTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelTEQ-impl-TRel-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-reflects-barbs} \( (\text{indRelTEQ} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{rel-with-target-impl-TRel-weakly-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \( \text{and} \ \text{Rel} :: ((\text{procS}, \text{procT}) \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc}) \text{set} \)
  \item \text{assumes reflection: rel-weakly-reflects-barbs} \( \text{Rel} (\text{STCalWB SWB TWB}) \)
  \item \text{and} \ targetInRel: \( \forall T1 T2 . (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \)
  \item \text{shows rel-weakly-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelRTPO-impl-TRel-weakly-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-weakly-reflects-barbs} \( (\text{indRelRTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-weakly-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelLTPO-impl-TRel-weakly-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-weakly-reflects-barbs} \( (\text{indRelLTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-weakly-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelTEQ-impl-TRel-weakly-reflects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes reflection: rel-weakly-reflects-barbs} \( (\text{indRelTEQ} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-weakly-reflects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

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

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelRTPO-impl-TRel-respects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes repection: rel-respects-barbs} \( (\text{indRelRTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-respects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelLTPO-impl-TRel-respects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes repection: rel-respects-barbs} \( (\text{indRelLTPO} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-respects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{proof}

\begin{proof}

\textbf{lemma} \text{(in encoding-wrt-barbs)} \text{indRelTEQ-impl-TRel-respects-barbs}:
\begin{itemize}
  \item \text{fixes} \( \text{TRel} :: (\text{procT} \times \text{procT}) \) \text{set}
  \item \text{assumes repection: rel-respects-barbs} \( (\text{indRelTEQ} \text{ TRel}) (\text{STCalWB SWB TWB}) \)
  \item \text{shows rel-respects-barbs} \( \text{TRel} \) \( \text{TWB} \)
\end{itemize}
\end{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) (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-rev:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: weak-reduction-simulation ((indRelRTPO TRel) (STCal Source Target)
shows weak-reduction-simulation \(((T\Rel^+)\cdot^1)\ \Target\)

(proof)

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

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

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

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

(proof)

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

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

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

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

\(\text{and } \text{target}: \forall T1 T2. \ (T1, T2) \in T\Rel \rightarrow (\Target\Term T1, \Target\Term T2) \in \text{Rel}\)

\(\text{and } \text{trel}: \forall T1 T2. \ (\Target\Term T1, \Target\Term T2) \in \text{Rel} \rightarrow (T1, T2) \in T\Rel^+\)

\(\text{shows } \text{weak-reduction-simulation } (T\Rel^+) \ \Target\)

(proof)

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

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

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

\(\text{shows } \text{weak-reduction-simulation } (T\Rel^+) \ \Target\)

(proof)

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

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

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

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

\(\text{and } \text{target}: \forall T1 T2. \ (T1, T2) \in T\Rel \rightarrow (\Target\Term T1, \Target\Term T2) \in \text{Rel}\)

\(\text{and } \text{trel}: \forall T1 T2. \ (\Target\Term T1, \Target\Term T2) \in \text{Rel} \rightarrow (T1, T2) \in T\Rel^+\)

\(\text{shows } \text{strong-reduction-simulation } (T\Rel^+) \ \Target\)

(proof)

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

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

\(\text{assumes } \text{sim}: \text{weak-reduction-simulation } \text{indRelRTPO } T\Rel \ (\text{STCal Source Target})\)

\(\text{shows } \text{strong-reduction-simulation } (T\Rel^+) \ \Target\)

(proof)

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

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

\(\text{assumes } \text{sim}: \text{weak-reduction-simulation } \text{indRelRTPO } T\Rel^{-1} \ (\text{STCal Source Target})\)

\(\text{shows } \text{strong-reduction-simulation } ((T\Rel^+)^{-1}) \ \Target\)

(proof)

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

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

\(\text{assumes } \text{sim}: \text{weak-reduction-simulation } \text{indRelLTPO } T\Rel \ (\text{STCal Source Target})\)

\(\text{shows } \text{strong-reduction-simulation } ((T\Rel^+)^{-1}) \ \Target\)

(proof)
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{strong-reduction-simulation} \ (\text{indRelLTOPO} \ TRel)^{-1}$ ($\text{STCal Source Target}$)
shows $\text{strong-reduction-simulation} \ ((TRel^\prime)^{-1}) \ \text{Target}$
(proof)

lemma (in encoding) $\text{rel-with-target-impl-refC-transC-TRel-is-strong-reduction-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
and $Rel :: (('procS, 'procT) Proc 
	imes ('procS, 'procT) Proc) set$
assumes sim: $\text{strong-reduction-simulation} \ Rel \ (\text{STCal Source Target})$
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^\prime$
shows $\text{strong-reduction-simulation} \ (TRel^\prime) \ \text{Target}$
(proof)

lemma (in encoding) $\text{indRelTEQ-impl-TRel-is-strong-reduction-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{strong-reduction-simulation} \ (\text{indRelTEQ} \ TRel) \ (\text{STCal Source Target})$
shows $\text{strong-reduction-simulation} \ (TRel^\prime) \ \text{Target}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelRTPO-impl-TRel-is-weak-barbed-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{weak-barbed-simulation} \ (\text{indRelRTPO} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{weak-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelLTOPO-impl-TRel-is-weak-barbed-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{weak-barbed-simulation} \ (\text{indRelLTOPO} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{weak-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelTEQ-impl-TRel-is-weak-barbed-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{weak-barbed-simulation} \ (\text{indRelTEQ} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{weak-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelRTPO-impl-TRel-is-strong-reduction-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{strong-barbed-simulation} \ (\text{indRelRTPO} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{strong-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelLTOPO-impl-TRel-is-strong-barbed-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{strong-barbed-simulation} \ (\text{indRelLTOPO} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{strong-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

lemma (in encoding-wrt-barbs) $\text{indRelTEQ-impl-TRel-is-strong-barbed-simulation}$:
fixes $TRel :: ('procT 
	imes 'procT) set$
assumes sim: $\text{strong-barbed-simulation} \ (\text{indRelTEQ} \ TRel) \ (\text{STCalWB SWB TWB})$
shows $\text{strong-barbed-simulation} \ (TRel^\prime) \ \text{TWB}$
(proof)

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

lemma (in encoding) $\text{rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation}$:
If indRelRTPO, indRelLTPO, or indRelTEQ is a coupled simulation then so is the corresponding target term relation.

**Lemma (in encoding)** indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-reduction-contrasimulation} \text{ Rel} (\text{STCal Source Target})$
- **Shows** weak-reduction-contrasimulation $(TRel^+) \text{ Target}$

**Lemma (in encoding)** indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-reduction-contrasimulation} \text{ (indRelLTPO TRel) (STCal Source Target)}$
- **Shows** weak-reduction-contrasimulation $(TRel^+) \text{ Target}$

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

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-reduction-contrasimulation} \text{ Rel} (\text{STCal Source Target})$
- **Shows** weak-reduction-contrasimulation $(TRel^+) \text{ Target}$

**Lemma (in encoding)** indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-reduction-contrasimulation} \text{ (indRelTEQ TRel) (STCal Source Target)}$
- **Shows** weak-reduction-contrasimulation $(TRel^+) \text{ Target}$

**Lemma (in encoding)** indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-barbed-contrasimulation} \text{ (indRelRTPO TRel) (STCalWB SWB TWB)}$
- **Shows** weak-barbed-contrasimulation $(TRel^+) \text{ TWB}$

**Lemma (in encoding)** indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-barbed-contrasimulation} \text{ (indRelLTPO TRel) (STCalWB SWB TWB)}$
- **Shows** weak-barbed-contrasimulation $(TRel^+) \text{ TWB}$

**Lemma (in encoding)** indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation:

- **Fixes** $TRel :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set})$
- **Assumes** $\text{conSim} :: \text{weak-barbed-contrasimulation} \text{ (indRelTEQ TRel) (STCalWB SWB TWB)}$
- **Shows** weak-barbed-contrasimulation $(TRel^+) \text{ TWB}$
lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes couSim: weak-reduction-coupled-simulation (indRelLTPO TRel) (STCal Source Target)
  shows weak-reduction-coupled-simulation (TRel⁺) Target
  ⟨proof⟩

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

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

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

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

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

lemma (in encoding) rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
  assumes corSim: weak-reduction-correspondence-simulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  and trel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel⁺
  shows weak-reduction-correspondence-simulation (TRel⁺) Target
  ⟨proof⟩

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

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

lemma (in encoding)
rel-with-target-impl-relC-transC-TRel-is-weak-reduction-correspondence-simulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
  assumes corSim: weak-reduction-correspondence-simulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
and \( trel: \forall T1 T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^* \)
shows weak-reduction-correspondence-simulation (\text{TRel}^*) \text{Target}
(proof)

lemma (in encoding) indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{corSim}: weak-reduction-correspondence-simulation (indRelTEQ \text{TRel}) (\text{STCal Source Target})
shows weak-reduction-correspondence-simulation (\text{TRel}^*) \text{Target}
(proof)

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-correspondence-simulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{corSim}: weak-barbed-correspondence-simulation (indRelRTPO \text{TRel}) (\text{STCalWB SWB TWB})
shows weak-barbed-correspondence-simulation (\text{TRel}^*) \text{TWB}
(proof)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-correspondence-simulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{corSim}: weak-barbed-correspondence-simulation (indRelLTPO \text{TRel}) (\text{STCalWB SWB TWB})
shows weak-barbed-correspondence-simulation (\text{TRel}^*) \text{TWB}
(proof)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-correspondence-simulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{corSim}: weak-barbed-correspondence-simulation (indRelTEQ \text{TRel}) (\text{STCalWB SWB TWB})
shows weak-barbed-correspondence-simulation (\text{TRel}^*) \text{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} × \text{'procT}) set
\text{and Rel} :: ((\text{'procS}, \text{'procT}) \text{Proc} × (\text{'procS}, \text{'procT}) \text{Proc}) set
assumes \text{bism}: weak-reduction-bisimulation \text{Rel} (\text{STCal Source Target})
\text{and target:} \forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}
\text{and trel:} \forall T1 T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+
shows weak-reduction-bisimulation (\text{TRel}^*) \text{Target}
(proof)

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-bisimulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{bism}: weak-reduction-bisimulation (indRelRTPO \text{TRel}) (\text{STCal Source Target})
shows weak-reduction-bisimulation (\text{TRel}^*) \text{Target}
(proof)

lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-bisimulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
assumes \text{bism}: weak-reduction-bisimulation (indRelLTPO \text{TRel}) (\text{STCal Source Target})
shows weak-reduction-bisimulation (\text{TRel}^*) \text{Target}
(proof)

lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation:
fixes \text{TRel} :: (\text{'procT} × \text{'procT}) set
\text{and Rel} :: ((\text{'procS}, \text{'procT}) \text{Proc} × (\text{'procS}, \text{'procT}) \text{Proc}) set
assumes \text{bism}: weak-reduction-bisimulation \text{Rel} (\text{STCal Source Target})
\text{and target:} \forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}
\text{and trel:} \forall T1 T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+
shows weak-reduction-bisimulation (\text{TRel}^*) \text{Target}
(proof)
lemma (in encoding) indRelTEQ-impl-TRel-is-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
  shows weak-reduction-bisimulation (TRel^+) Target
  ⟨proof⟩

lemma (in encoding) rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
  assumes bisim: strong-reduction-bisimulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  and trel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel^*
  shows strong-reduction-bisimulation (TRel^+) Target
  ⟨proof⟩

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

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

lemma (in encoding) rel-with-target-impl-refC-transC-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
  assumes bisim: strong-reduction-bisimulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  and trel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel^*
  shows strong-reduction-bisimulation (TRel^+) Target
  ⟨proof⟩

lemma (in encoding) indRelTEQ-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: weak-barbed-bisimulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows weak-barbed-bisimulation (TRel^+) TWB
  ⟨proof⟩

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

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

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes bisim: weak-barbed-bisimulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows weak-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**

\[
\text{fixes } TRel :: \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelRTPO} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^+)\) \(\text{TWB}\)

**proof**

**lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-strong-barbed-bisimulation:**

\[
\text{fixes } TRel :: \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelLTPO} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^+)\) \(\text{TWB}\)

**proof**

**lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-strong-barbed-bisimulation:**

\[
\text{fixes } TRel :: \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelTEQ} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^*)\) \(\text{TWB}\)

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**

\[
\text{fixes } TRel :: \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelRTPO} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^+)\) \(\text{TWB}\)

**proof**

**abbreviation (in encoding) indRelRSTinfix**

\[
\text{def} \quad \text{encR}: \text{('SourceTerm S, TargetTerm ([S])} \in \text{indRelRST SRel TRel)}
\]

**source:** \((S_1, S_2) \in \text{SRel} \implies \text{('SourceTerm S_1, SourceTerm S_2} \in \text{indRelRST SRel TRel)}

**target:** \((T_1, T_2) \in \text{TRel} \implies \text{('TargetTerm T_1, TargetTerm T_2} \in \text{indRelRST SRel TRel)}

**abbreviation (in encoding) indRelRSTimplinfix**

\[
\text{def} \quad \text{encR}: \text{('proc}S \times \text{'proc}S) \text{ set} \implies \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelRTPO} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^+)\) \(\text{TWB}\)

**proof**

**abbreviation (in encoding) indRelRSTimplinfix**

\[
\text{def} \quad \text{encR}: \text{('proc}S \times \text{'proc}S) \text{ set} \implies \text{('proc}T \times \text{'proc}T) \text{ set}
\]

**assumes** bisim: strong-barbed-bisimulation \((\text{indRelRTPO} \ TRel) (\text{STCalWB SWB TWB})\)

**shows** strong-barbed-bisimulation \((TRel^+)\) \(\text{TWB}\)

**proof**

**lemma (in encoding) indRelRSTrefl:**

**fixes** SRel :: \((\text{'proc}S \times \text{'proc}S) \text{ set}

**and** TRel :: \((\text{'proc}T \times \text{'proc}T) \text{ set}

48
assumes reflS: refl SRel
    and reflT: refl TRel
shows refl (indRelRSTPO SRel TRel)
  ⟨proof⟩

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

lemma (in encoding) refl-trans-closure-of-indRelRST:
  fixes SRel :: (′procS × ′procS) set
    and TRel :: (′procT × ′procT) set
assumes reflS: refl SRel
    and reflT: refl TRel
shows indRelRSTPO SRel TRel = (indRelRST SRel TRel)*
  ⟨proof⟩

inductive-set (in encoding) indRelLST
  :: (′procS × ′procS) set ⇒ (′procT × ′procT) set
    ⇒ (((′procS, ′procT) Proc) × ((′procS, ′procT) Proc)) set
for SRel :: (′procS × ′procS) set
    and TRel :: (′procT × ′procT) set
where
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelLST SRel TRel | source: (S1, S2) ∈ SRel ⇒ (SourceTerm S1, SourceTerm S2) ∈ indRelLST SRel TRel | target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelLST SRel TRel
abbreviation (in encoding) indRelLSTinf
  :: (′procS, ′procT) Proc ⇒ (′procS × ′procS) set ⇒ (′procT × ′procT) set
    ⇒ (′procS, ′procT) Proc ⇒ bool (¬ R[′]L<-,> - [75, 75, 75, 75] 80)
where
  P R[′]L<,>Q ≡ (P, Q) ∈ indRelLST SRel TRel

inductive-set (in encoding) indRelLSTPO
  :: (′procS × ′procS) set ⇒ (′procT × ′procT) set
    ⇒ (((′procS, ′procT) Proc) × ((′procS, ′procT) Proc)) set
for SRel :: (′procS × ′procS) set
    and TRel :: (′procT × ′procT) set
where
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelLSTPO SRel TRel | source: (S1, S2) ∈ SRel ⇒ (SourceTerm S1, SourceTerm S2) ∈ indRelLSTPO SRel TRel | target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelLSTPO SRel TRel | trans: [(P, Q) ∈ indRelLSTPO SRel TRel; (Q, R) ∈ indRelLSTPO SRel TRel] ⇒ (P, R) ∈ indRelLSTPO SRel TRel
abbreviation (in encoding) indRelLSTPOinf
  :: (′procS, ′procT) Proc ⇒ (′procS × ′procS) set ⇒ (′procT × ′procT) set
    ⇒ (′procS, ′procT) Proc ⇒ bool (¬ L<-,>- [75, 75, 75, 75] 80)
where
  P L<,>Q ≡ (P, Q) ∈ indRelLSTPO SRel TRel

lemma (in encoding) indRelLSTPO-refl:
  fixes SRel :: (′procS × ′procS) set
    and TRel :: (′procT × ′procT) set
assumes reflS: refl SRel
    and reflT: refl TRel
shows refl (indRelLSTPO SRel TRel)
  ⟨proof⟩
lemma (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) indRelST
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set
  for SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  where
  encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelST SRel TRel |
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelST SRel TRel |
  source: (S1, S2) ∈ SRel ⇒ (SourceTerm S1, SourceTerm S2) ∈ indRelST SRel TRel |
  target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelST SRel TRel

abbreviation (in encoding) indRelSTinfix
  :: ('procS × 'procT) Proc ⇒ ('procS × 'procS) set ⇒ ('procT × 'procT) set
  for SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  where
  P R[,]<SRel,TRel> Q ≡ (P, Q) ∈ indRelST SRel TRel

lemma (in encoding) indRelST-symm:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes symmS: symm SRel
  and symmT: symm TRel
  shows symm (indRelST 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
  for SRel :: ('procS × 'procS) set
  and TRel :: ('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
assumes reflT: refl TRel
shows refl (indRelSTEQ SRel TRel)
⟨proof⟩

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

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

lemma (in encoding) refl-trans-closure-of-indRelST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
shows indRelSTEQ SRel TRel = (indRelST SRel TRel)∗
⟨proof⟩

lemma (in encoding) refl-symm-trans-closure-of-indRelST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
and symmS: sym SRel
and symmT: sym TRel
shows indRelSTEQ SRel TRel = (symcl ((indRelST SRel TRel)∗))∗
⟨proof⟩

lemma (in encoding) symm-closure-of-indRelRST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
and symmS: sym SRel
and symmT: sym TRel
shows indRelST SRel TRel = symcl (indRelRST SRel TRel)
and indRelSTEQ SRel TRel = (symcl ((indRelRST SRel TRel)∗))∗
⟨proof⟩

lemma (in encoding) symm-closure-of-indRelLST:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes reflT: refl TRel
and symmS: sym SRel
and symmT: sym TRel
shows indRelST SRel TRel = symcl (indRelLST SRel TRel)
and indRelSTEQ SRel TRel = (symcl ((indRelLST SRel TRel)∗))∗
⟨proof⟩

lemma (in encoding) symm-trans-closure-of-indRelRSTPO:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes symmS: sym SRel
and symmT: sym TRel
shows indRelSTEQ SRel TRel = (symcl (indRelRSTPO SRel TRel))∗
⟨proof⟩
lemma (in encoding) symm-trans-closure-of-indRelLSTPO:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes symmS: sym SRel
  and symmT: sym TRel
  shows indRelSTEQ SRel TRel = (symcl (indRelLSTPO SRel TRel)) +
(proof)

If the relations indRelRST, indRelLST, or indRelST contain a pair of target terms, then this pair is also related by the considered target term relation. Similarly a pair of source terms is related by the considered source term relation.

lemma (in encoding) indRelRST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP R[ ][ ] R< SRel, TRel > SourceTerm SQ
  shows (SP, SQ) ∈ SRel
(proof)

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

lemma (in encoding) indRelLST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP R[ ][ ] L< SRel, TRel > SourceTerm SQ
  shows (SP, SQ) ∈ SRel
(proof)

lemma (in encoding) indRelLST-to-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  assumes rel: TargetTerm TP R[ ][ ] L< SRel, TRel > TargetTerm TQ
  shows (TP, TQ) ∈ TRel
(proof)

lemma (in encoding) indRelST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP R[ ][ ] < SRel, TRel > SourceTerm SQ
  shows (SP, SQ) ∈ SRel
(proof)

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

If the relations indRelRSTPO or indRelLSTPO contain a pair of target terms, then this pair is also
related by the transitive closure of the considered target term relation. Similarly a pair of source terms is related by the transitive closure of the source term relation. A pair of a source and a target term results from the combination of pairs in the source relation, the target relation, and the encoding function. Note that, because of the symmetry, no similar condition holds for indRelSTEQ.

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

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( TRel :: (\langle procT \times \langle procT \rangle \rangle set \)
- **and** \( P Q :: (\langle procS, \langle procT \rangle \rangle Proc \)

**Assumes** \( P \leq SRel, TRel > Q \)

**Shows**

\[ \forall SP SQ, SP \in S P \land SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+ \]
\[ \forall SP TQ, SP \in S P \land TQ \in T Q \longrightarrow (\exists S. (SP, S) \in SRel^+ \land ([S], TQ) \in TRel^+) \]
\[ \forall TP SQ, TP \in T P \land SQ \in S Q \longrightarrow False \]
\[ \forall TP TQ, TP \in T P \land TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+ \]

\( \langle proof \rangle \)

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

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( TRel :: (\langle procT \times \langle procT \rangle \rangle set \)
- **and** \( P Q :: (\langle procS, \langle procT \rangle \rangle Proc \)

**Assumes** \( P \leq SRel, TRel > Q \)

**Shows**

\[ \forall SP SQ, SP \in S P \land SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+ \]
\[ \forall SP TQ, SP \in S P \land TQ \in T Q \longrightarrow False \]
\[ \forall TP SQ, TP \in T P \land SQ \in S Q \longrightarrow (\exists S. (TP, [S]) \in TRel^+ \land (S, SQ) \in SRel^+) \]
\[ \forall TP TQ, TP \in T P \land TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+ \]

\( \langle proof \rangle \)

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

**Lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-preserves-barbs:**

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( Rel :: (\langle procS, \langle procT \rangle \rangle Proc \times (\langle procS, \langle procT \rangle \rangle Proc \rangle set \)

**Assumes** \( \text{preservation: rel-preserves-barbs Rel (STCalWB SWB TWB) \)
- **and** \( \text{sourceInRel: } \forall S1 S2. (S1, S2) \in SRel \longrightarrow (SourceTerm S1, SourceTerm S2) \in Rel \)

**Shows** \( \text{rel-preserves-barbs SRel SWB} \)

\( \langle proof \rangle \)

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

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( TRel :: (\langle procT \times \langle procT \rangle \rangle set \)

**Assumes** \( \text{preservation: rel-preserves-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB) \)

**Shows** \( \text{rel-preserves-barbs SRel SWB} \)
- **and** \( \text{rel-preserves-barbs TRel TWB} \)

\( \langle proof \rangle \)

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

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( TRel :: (\langle procT \times \langle procT \rangle \rangle set \)

**Assumes** \( \text{preservation: rel-preserves-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB) \)

**Shows** \( \text{rel-preserves-barbs SRel SWB} \)
- **and** \( \text{rel-preserves-barbs TRel TWB} \)

\( \langle proof \rangle \)

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

- **Fixes** \( SRel :: (\langle procS \times \langle procS \rangle \rangle set \)
- **and** \( TRel :: (\langle procT \times \langle procT \rangle \rangle set \)

**Assumes** \( \text{preservation: rel-preserves-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB) \)

**Shows** \( \text{rel-preserves-barbs SRel SWB} \)
- **and** \( \text{rel-preserves-barbs TRel TWB} \)

\( \langle proof \rangle \)
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-preserve-barbs:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-barbs SRel SWB
  and rel-weakly-preserves-barbs TRel TWB
(proof)

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

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

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

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

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
\[
\begin{align*}
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes reflection: } & \text{rel-reflects-barbs (indRelSTEQ SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-reflects-barbs SRel SWB} \\
\text{and } & \text{rel-reflects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{rel-with-source-impl-SRel-weakly-reflects-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{Rel} :: (\text{procS} \times \text{procT}) \text{ Proc} \times (\text{procS} \times \text{procT}) \text{ Proc} \text{ set} \\
\text{assumes reflection: } & \text{rel-weakly-reflects-barbs Rel (STCalWB SWB TWB)} \\
\text{and } & \text{sourceInRel: } \forall S_1 S_2. (S_1, S_2) \in \text{SRel} \rightarrow (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel} \\
\text{shows } & \text{rel-weakly-reflects-barbs SRel SWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes reflection: } & \text{rel-weakly-reflects-barbs (indRelRSTPO SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-weakly-reflects-barbs SRel SWB} \\
\text{and } & \text{rel-weakly-reflects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes reflection: } & \text{rel-weakly-reflects-barbs (indRelLSTPO SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-weakly-reflects-barbs SRel SWB} \\
\text{and } & \text{rel-weakly-reflects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes reflection: } & \text{rel-weakly-reflects-barbs (indRelSTEQ SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-weakly-reflects-barbs SRel SWB} \\
\text{and } & \text{rel-weakly-reflects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

If \(\text{indRelRSTPO}, \text{indRelLSTPO}, \) or \(\text{indRelSTPO}\) respects barbs then so do the corresponding source term and target term relations.

**Lemma (in encoding-wrt-barbs)** \(\text{indRelRSTPO-impl-SRel-and-TRel-respect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes respectation: } & \text{rel-respects-barbs (indRelRSTPO SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-respects-barbs SRel SWB} \\
\text{and } & \text{rel-respects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{indRelLSTPO-impl-SRel-and-TRel-respect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes respectation: } & \text{rel-respects-barbs (indRelLSTPO SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-respects-barbs SRel SWB} \\
\text{and } & \text{rel-respects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]

**Lemma (in encoding-wrt-barbs)** \(\text{indRelSTEQ-impl-SRel-and-TRel-respect-barbs:}\)
\[
\begin{align*}
\text{fixes } & \text{SRel} :: (\text{procS} \times \text{procS}) \text{ set} \\
\text{and } & \text{TRel} :: (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes respectation: } & \text{rel-respects-barbs (indRelSTEQ SRel TRel)} \text{ (STCalWB SWB TWB)} \\
\text{shows } & \text{rel-respects-barbs SRel SWB} \\
\text{and } & \text{rel-respects-barbs TRel TWB} \\
(\text{proof}) \\
\end{align*}
\]
fixes \( SRel \) :: \( \langle \text{procS} \times \text{procS} \rangle \) set
and \( TRel \) :: \( \langle \text{procT} \times \text{procT} \rangle \) set
assumes respection: rel-weakly-respects-barbs \((\text{indRelRSTPO} SRel TRel)\) \((\text{STCalWB SWB TWB})\)
shows rel-weakly-respects-barbs \( SRel \) SWB
and rel-weakly-respects-barbs \( TRel \) TWB
⟨proof⟩

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

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

If \( TRel \) is reflexive then \( \text{indRelRTPO} \) is a subrelation of \( \text{indRelTEQ} \). If \( SRel \) is reflexive then \( \text{indRelRTPO} \) is a subrelation of \( \text{indRelRTPO} \). Moreover, \( \text{indRelRSTPO} \) is a subrelation of \( \text{indRelSTEQ} \).

lemma (in encoding) \( \text{indRelRTPO-to-indRelTEQ} \):
fixes \( TRel \) :: \( \langle \text{procT} \times \text{procT} \rangle \) set
and \( P Q \) :: \( \langle \text{procS}, \text{procT} \rangle \) Proc
assumes rel: \( P \preceq_{\text{RT}} <\text{TRel}> Q \)
and reflT: refl \( TRel \)
sows \( P \sim_T <\text{TRel}> Q \)
⟨proof⟩

lemma (in encoding) \( \text{indRelRTPO-to-indRelRSTPO} \):
fixes \( SRel \) :: \( \langle \text{procS} \times \text{procS} \rangle \) set
and \( TRel \) :: \( \langle \text{procT} \times \text{procT} \rangle \) set
and \( P Q \) :: \( \langle \text{procS}, \text{procT} \rangle \) Proc
assumes rel: \( P \preceq_{\text{RT}} <\text{TRel}> Q \)
and reflS: refl \( SRel \)
sows \( P \preceq_{\text{R}} <\text{SRel},\text{TRel}> Q \)
⟨proof⟩

If \( \text{indRelRTPO} \) is a bisimulation and \( SRel \) is a reflexive bisimulation then also \( \text{indRelRSTPO} \) is a bisimulation.

lemma (in encoding) \( \text{indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation} \):
fixes \( SRel \) :: \( \langle \text{procS} \times \text{procS} \rangle \) set
and \( TRel \) :: \( \langle \text{procT} \times \text{procT} \rangle \) set
assumes \( \text{bisim}_T \): weak-reduction-bisimulation \((\text{indRelRTPO \ TRel}) \) \((\text{STCal Source Target})\)

and \( \text{bisim}_S \): weak-reduction-bisimulation \(\text{SRel Source}\)

and \( \text{refl}_S \): refl \(\text{SRel}\)

shows weak-reduction-bisimulation \(\text{(indRelRSTPO \ SRel \ TRel}) \) \((\text{STCal Source Target})\)

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

end

theory SuccessSensitiveness
imports SourceTargetRelation
begin

6 Success Sensitiveness and Barbs

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

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

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

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

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

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

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

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

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

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

An encoding strongly respects some set of barbs if, for every source term \( S \) and all considered barbs \( a \), \( S \) has a iff the translation of \( S \) has a.

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

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

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

**Shows** enc-preserves-barbs = (\(\forall \text{ Barbs} \ . \text{ enc-preserves-barb-set Barbs} \))

(\(\text{proof}\))

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

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

**Abbreviation (in encoding-wrt-barbs) enc-reflects-barbs :: bool where**

enc-reflects-barbs ≡ enc-reflects-binary-pred (\(\lambda \ P \ . \ P \downarrow \))

**Lemma (in encoding-wrt-barbs) enc-reflects-barbs-and-barb-set:**

**Shows** enc-reflects-barbs = (\(\forall \text{ Barbs} \ . \text{ enc-reflects-barb-set Barbs} \))

(\(\text{proof}\))

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

enc-respects-barb-set Barbs ≡ enc-preserves-barb-set Barbs \land enc-reflects-barb-set Barbs

**Lemma (in encoding-wrt-barbs) enc-respects-barbs-and-barb-set:**

**Shows** enc-respects-barbs = (\(\forall \text{ Barbs} \ . \text{ enc-respects-barb-set Barbs} \))

(\(\text{proof}\))

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

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

**Fixes** Barbs :: 'barbs set

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

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

(\(\text{proof}\))

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

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

(\(\text{proof}\))

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

**Fixes** Barbs :: 'barbs set

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

(\(\text{proof}\))

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

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

(\(\text{proof}\))

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

Accordingly an encoding is success sensitive iff there exists such a relation between source and target terms that weakly respects the barb success.
lemma (in encoding-wrt-barbs) success-sensitive-cond:
  fixes success :: 'barbs
  shows enc-weakly-respects-barb-set {success} = (\forall S. S \Downarrow SWB > success \iff [S] \Downarrow TWB > success)
  ⟨proof⟩

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

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 (Source)\omega \mapsto [S] \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 (Target)\omega \mapsto S \mapsto (Source)\omega)
  ⟨proof⟩

abbreviation rel-preserves-divergence :: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
  where
  rel-preserves-divergence Rel Cal ≡ rel-preserves-pred Rel (\lambda P. P \mapsto (Cal)\omega)

abbreviation rel-reflects-divergence :: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
  where
  rel-reflects-divergence Rel Cal ≡ rel-reflects-pred Rel (\lambda P. P \mapsto (Cal)\omega)

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

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

60
lemma (in encoding) divergence-respection-cond:
shows enc-respects-divergence = (∀ S. [S] \rightarrow (Target)ω \leftrightarrow S \rightarrow (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 \rightarrow (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-preserves-divergence:
shows enc-reflects-divergence = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ rel-preserves-divergence Rel (STCal Source Target))
⟨proof⟩

lemma (in encoding) divergence-reflection-iff-source-target-rel-reflects-divergence:
shows enc-reflects-divergence = (∃ Rel. (∀ S. (TargetTerm ([S]), SourceTerm S) ∈ Rel)
∧ rel-reflects-divergence Rel (STCal Source Target))
⟨proof⟩
An encoding respects divergence iff there exists a relation that relates source terms and their literal
translations in both directions and respects divergence.

lemma (in encoding) divergence-respection-iff-source-target-rel-respects-divergence:
shows enc-respects-divergence = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ rel-respects-divergence Rel (STCal Source Target))
and enc-respects-divergence = (∃ Rel.
(∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
∧ rel-respects-divergence Rel (STCal Source Target))
⟨proof⟩

end
theory OperationalCorrespondence
imports SourceTargetRelation
begin

8 Operational Correspondence

We consider different variants of operational correspondence. This criterion consists of a completeness
and a soundness condition and is often defined with respect to a relation TRel on target terms.
Operational completeness modulo TRel ensures that an encoding preserves source term behaviour modulo TRel by requiring that each sequence of source term steps can be mimicked by its translation such that the respective derivatives are related by TRel.

\textbf{abbreviation (in encoding) operational-complete} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
operational-complete \( TRel \equiv \forall S S'. S \mapsto \text{Source* } S' \mapsto (\exists T. [[S]] \mapsto \text{Target* } T \land ([S'], T) \in TRel) \)

We call an encoding strongly operational complete modulo TRel if each source term step has to be mimicked by single target term step of its translation.

\textbf{abbreviation (in encoding) strongly-operational-complete} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
strongly-operational-complete \( TRel \equiv \forall S S'. S \mapsto \text{Source* } S' \mapsto (\exists T. [[S]] \mapsto \text{Target* } T \land ([S'], T) \in TRel) \)

Operational soundness ensures that the encoding does not introduce new behaviour. An encoding is weakly operational sound modulo TRel if each sequence of target term steps is part of the translation of a sequence of source term steps such that the derivatives are related by TRel. It allows for intermediate states on the translation of source term step that are not the result of translating a source term.

\textbf{abbreviation (in encoding) weakly-operational-sound} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
weakly-operational-sound \( TRel \equiv \forall S T. [[S]] \mapsto \text{Target* } T \mapsto (\exists S'T'. S \mapsto \text{Source* } S' \land T \mapsto \text{Target* } T' \land ([S'], T') \in TRel) \)

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

\textbf{abbreviation (in encoding) operational-sound} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
operational-sound \( TRel \equiv \forall S T. [[S]] \mapsto \text{Target* } T \mapsto (\exists S'. S \mapsto \text{Source* } S' \land ([S'], T) \in TRel) \)

Strong operational soundness modulo TRel is a stricter variant of operational soundness, where a single target term step has to be mapped on a single source term step.

\textbf{abbreviation (in encoding) strongly-operational-sound} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
strongly-operational-sound \( TRel \equiv \forall S T. [[S]] \mapsto \text{Target* } T \mapsto (\exists S'. S \mapsto \text{Source* } S' \land ([S'], T) \in TRel) \)

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

\textbf{abbreviation (in encoding) weakly-operational-corresponding} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
weakly-operational-corresponding \( TRel \equiv \) operational-complete \( TRel \land \) weakly-operational-sound \( TRel \)

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

\textbf{abbreviation (in encoding) operational-corresponding} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
operational-corresponding \( TRel \equiv \) operational-complete \( TRel \land \) operational-sound \( TRel \)

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

\textbf{abbreviation (in encoding) strongly-operational-corresponding} :: \( ('procT \times 'procT) \text{ set } \Rightarrow \text{bool} \) \text{ where} 
strongly-operational-corresponding \( TRel \equiv \) strongly-operational-complete \( TRel \land \) strongly-operational-sound \( TRel \)
8.1 Trivial Operational Correspondence Results

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

**Lemma (in encoding) operational-correspondence-modulo-all-relation:**
- shows operational-complete \{(T1, T2). True\}
- and weakly-operational-sound \{(T1, T2). True\}
- and operational-sound \{(T1, T2). True\}

**Proof**

**Lemma all-relation-is-weak-reduction-bisimulation:**
- fixes Cal :: 'a processCalculation
- shows weak-reduction-bisimulation \{(a, b). 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 \(\exists \text{TRel. strongly-operational-corresponding TRel}\)
- \(= (\forall S. (\exists S'. S \rightarrow Source S') \leftrightarrow (\exists T. [S] \rightarrow Target T))\)
- and \(\exists \text{TRel. strongly-operational-corresponding TRel}\)
- \(\land \text{weak-reduction-bisimulation TRel Target}\)
- \(= (\forall S. (\exists S'. S \rightarrow Source S') \leftrightarrow (\exists T. [S] \rightarrow Target T))\)

**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 \(\text{indRelRTPO}\), 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 :: \(\langle ('\text{procS}', '\text{procT}') \text{ Proc} \times ('\text{procS}', '\text{procT}') \text{ Proc} \rangle \) set
- and TRel :: \(\langle '\text{procT} \times '\text{procT} \rangle \) set
- assumes A1: \(\forall S. (\langle SourceTerm S, TargetTerm ([S]) \rangle) \in \text{Rel}\)
- and A2: \(\forall S T. (\langle SourceTerm S, TargetTerm T \rangle) \in \text{Rel} \rightarrow (\langle [S], T \rangle) \in \text{TRel}^*\)
- and A3: \(\text{weak-reduction-simulation Rel (STCal Source Target)}\)
- shows operational-complete (TRel*)

**Proof**

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

**Proof**

**Lemma (in encoding) OCom-iff-weak-reduction-simulation:**
- fixes TRel :: \(\langle '\text{procT} \times '\text{procT} \rangle \) set
- shows (operational-complete (TRel*))
  - \& weak-reduction-simulation (TRel+) Target
  - = (\exists \text{Rel. (\forall S. (SourceTerm S, TargetTerm ([S]) \rangle) \in \text{Rel}})
  - \& (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (TargetTerm T1, TargetTerm T2) \in \text{Rel})
  - \& (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^*)
\[ \forall S \ T. \ (\text{SourceTerm } S, \ \text{TargetTerm } T) \in \text{Rel} \implies ([S], T) \in \text{TRel}^* \]
\[ \text{weak-reduction-simulation} \ \text{Rel} (\text{STCal Source Target}) \]

(proof)

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

**lemma (in encoding) SOCom-iff-indRelRTPO-is-strong-reduction-simulation:**

- fixes TRel :: ('procT × 'procT) set
- shows (strongly-operational-complete (TRel*)
  \[ \implies \text{strong-reduction-simulation} \ (\text{TRel}^*) \ ]
  \[ \text{Target} \]

(proof)

**lemma (in encoding) STCal-from-source-target-relation:**

- fixes TRel :: ('procT × 'procT) set
- shows (strongly-operational-complete (TRel*)
  \[ \implies \text{strong-reduction-simulation} \ (\text{TRel}^*) \ ]
  \[ \text{Target} \]

(proof)

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

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

(proof)

8.3 Weak Operational Soundness vs Contrasimulation

If the inverse of a relation that includes TRel and relates source terms and their literal translations is a contrasimulation, then the encoding is weakly operational sound.

**lemma (in encoding) weak-reduction-contrasimulation-impl-WOSou:**

- fixes Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
- and TRel :: ('procT × 'procT) set
- assumes A1: \( \forall S. \ (\text{SourceTerm } S, \ \text{TargetTerm } ([S])) \in \text{Rel} \)
and A2: \( \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^* \)
and A3: weak-reduction-contrasimulation \((\text{Rel}^{-1})\) \((\text{STCal Source Target})\)
shows weakly-operational-sound \((\text{TRel}^*)\)
(proof)

8.4 (Strong) Operational Soundness vs (Strong) Simulation

An encoding is operational sound modulo a relation \(\text{TRel}\) whose inverse is a weak reduction simulation on target terms iff there is a relation, like \(\text{indRelRTPO}\), that relates at least all source terms to their literal translations, includes \(\text{TRel}\), and whose inverse is a weak simulation.

lemma (in encoding) weak-reduction-simulation-impl-OSou:
fixes \text{Rel} :: \((\text{procS}, \text{procT})\) \text{Proc} \times \((\text{procS}, \text{procT})\) \text{Proc} \) set
and \text{TRel} :: \((\text{procT} \times \text{procT})\) set
assumes A1: \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
and A2: \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^* \)
and A3: weak-reduction-simulation \((\text{Rel}^{-1})\) \((\text{STCal Source Target})\)
shows operational-sound \((\text{TRel}^*)\)
(proof)

lemma (in encoding) OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation:
fixes \text{TRel} :: \((\text{procT} \times \text{procT})\) set
shows (operational-sound \((\text{TRel}^*)\))
\begin{align*}
& \land \text{weak-reduction-simulation} \((\text{TRel}^+)^{-1}\) Target \\
= \text{weak-reduction-simulation} \(((\text{indRelRTPO} \text{TRel})^{-1})\) \((\text{STCal Source Target})\)
\end{align*}
(proof)

lemma (in encoding) OSou-iff-weak-reduction-simulation:
fixes \text{TRel} :: \((\text{procT} \times \text{procT})\) set
shows (operational-sound \((\text{TRel}^*)\))
\begin{align*}
& \land \text{weak-reduction-simulation} \((\text{TRel}^+)^{-1}\) Target \\
= \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \\
& \land (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \\
& \land (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+ ) \\
& \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^* \)
\end{align*}
\(\land \text{weak-reduction-simulation} \((\text{Rel}^{-1})\) \((\text{STCal Source Target})\)\)
(proof)

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

lemma (in encoding) strong-reduction-simulation-impl-SOSou:
fixes \text{Rel} :: \((\text{procS}, \text{procT})\) \text{Proc} \times \((\text{procS}, \text{procT})\) \text{Proc} \) set
and \text{TRel} :: \((\text{procT} \times \text{procT})\) set
assumes A1: \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
and A2: \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^* \)
and A3: strong-reduction-simulation \((\text{Rel}^{-1})\) \((\text{STCal Source Target})\)
shows strongly-operational-sound \((\text{TRel}^*)\)
(proof)

lemma (in encoding) SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation:
fixes \text{TRel} :: \((\text{procT} \times \text{procT})\) set
shows (strongly-operational-sound \((\text{TRel}^*)\))
\begin{align*}
& \land \text{strong-reduction-simulation} \((\text{TRel}^+)^{-1}\) Target \\
= \text{strong-reduction-simulation} \(((\text{indRelRTPO} \text{TRel})^{-1})\) \((\text{STCal Source Target})\)
\end{align*}
(proof)

lemma (in encoding) SOSou-iff-strong-reduction-simulation:
fixes \text{TRel} :: \((\text{procT} \times \text{procT})\) set
shows (strongly-operational-sound \((\text{TRel}^*)\) \land \text{strong-reduction-simulation} \((\text{TRel}^+)^{-1}\) Target)
\[(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \land (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \land (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{TRel} \rightarrow (T1, T2) \in \text{TRel}^+)) \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+) \land \text{strong-reduction-simulation} (\text{Rel}^{-1}) (\text{STCal Source Target})\]

(proof)

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

\[
\begin{align*}
&\text{shows} (3 \text{TRel. strongly-operational-sound (TRel')} \\
&\land \text{strong-reduction-simulation} ((\text{TRel'})^{-1}) \text{Target} \\
&= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \\
&\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}'^+) \\
&\land \text{strong-reduction-simulation} (\text{Rel}^{-1}) (\text{STCal Source Target}))
\end{align*}
\]

(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:

\[
\begin{align*}
&\text{fixes Rel : } ((\text{procS}, \text{procT}) \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc}) \text{ set} \\
&\text{and TRel : } (\text{procT} \times \text{procT}) \text{ set} \\
&\text{assumes enc: } \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \\
&\text{and TRel: } (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+) \\
&\text{and cs: } \text{weak-reduction-correspondence-simulation} \text{Rel (STCal Source Target)} \\
&\text{shows weakly-operational-corresponding (TRel')}
\end{align*}
\]

(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:

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

(proof)

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

\[
\begin{align*}
&\text{fixes TRel : } (\text{procT} \times \text{procT}) \text{ set} \\
&\text{shows } (\text{weakly-operational-corresponding (TRel'}) \\
&\land \text{weak-reduction-correspondence-simulation (TRel') Target} \\
&= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \\
&\land (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \\
&\land (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{TRel}^{-1} \rightarrow (T1, T2) \in \text{TRel}^+) \\
&\land \text{weak-reduction-correspondence-simulation} \text{Rel (STCal Source Target))}
\end{align*}
\]

(proof)

**Lemma rel-includes-TRel-modulo-preorder:**

\[
\begin{align*}
&\text{fixes Rel : } ((\text{procS}, \text{procT}) \text{Proc} \times (\text{procS}, \text{procT}) \text{Proc}) \text{ set} \\
&\text{and TRel : } (\text{procT} \times \text{procT}) \text{ set} \\
&\text{assumes transT: trans TRel} \\
&\text{shows } ((\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \\
&\land (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{TRel}^{-1} \rightarrow (T1, T2) \in \text{TRel}^+) \\
&= \{ (T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \}
\end{align*}
\]

(proof)
lemma (in encoding) WOC-wrt-preorder-iff-reduction-correspondence-simulation:
  fixes TRel :: ('procT × 'procT) set
  shows (weakly-operational-corresponding TRel ∧ preorder TRel
       ∧ weak-reduction-correspondence-simulation TRel Target)
       = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
           ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
           ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
           ∧ preorder Rel
           ∧ weak-reduction-correspondence-simulation Rel (STCal Source Target))
⟨proof⟩

8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity

An encoding is operational corresponding w.r.t a weak bisimulation on target terms TRel iff there exists a relation, like indRelRTPO, that relates at least all source terms and their literal translations, includes TRel, and is a weak bisimulation. Thus this variant of operational correspondence ensures that source terms and their translations are weak bisimilar.

lemma (in encoding) OC-iff-indRelRTPO-is-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  shows (operational-corresponding (TRel⋆)
       ∧ weak-reduction-bisimulation (TRel†) Target)
       = weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
⟨proof⟩

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

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

lemma (in encoding) OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes eqT: equivalence TRel
  shows (operational-corresponding TRel ∧ weak-reduction-bisimulation TRel Target) ←→
       weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
⟨proof⟩

lemma (in encoding) OC-wrt-equivalence-iff-weak-reduction-bisimulation:
  fixes TRel :: ('procT × 'procT) set
  assumes eqT: equivalence TRel
  shows (operational-corresponding TRel ∧ weak-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 ∧ weak-reduction-bisimulation Rel (STCal Source Target))
⟨proof⟩
An encoding is strong operational corresponding w.r.t a strong bisimulation on target terms TRel iff there exists a relation, like indRelRTPO, that relates at least all source terms and their literal translations, includes TRel, and is a strong bisimulation. Thus this variant of operational correspondence ensures that source terms and their translations are strong bisimilar.

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

*fixes* TRel :: ('procT × 'procT) set

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

∧ strong-reduction-bisimulation (TRel*) Target

= strong-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

(proof)

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

*fixes* TRel :: ('procT × 'procT) set

*shows* (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)

∧ strong-reduction-bisimulation Rel (STCal Source Target))

(proof)

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

*fixes* TRel :: ('procT × 'procT) set

*shows* (strongly-operational-corresponding (TRel) ∧ preorder TRel)

∧ strong-reduction-bisimulation TRel Target

= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]]) ∈ Rel)

∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}

∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel −→ ([S], T) ∈ TRel)

∧ preorder Rel)

∧ strong-reduction-bisimulation Rel (STCal Source Target))

(proof)

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

*shows* (∃ TRel. strongly-operational-corresponding (TRel*)

∧ strong-reduction-bisimulation (TRel*) Target)

= (∃ Rel. (∀ (SourceTerm S, TargetTerm ([S]))) ∈ Rel)

∧ (∀ (SourceTerm S, TargetTerm T) ∈ Rel −→ (TargetTerm ([S]), TargetTerm T) ∈ Rel)

∧ strong-reduction-bisimulation Rel (STCal Source Target))

(proof)

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

*fixes* TRel :: ('procT × 'procT) set

*assumes* eqT: equivalence TRel

*shows* (strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target)

↔ strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)

(proof)

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

*fixes* TRel :: ('procT × 'procT) set

*assumes* eqT: equivalence TRel

*shows* (strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target)

↔ (∃ Rel.

(∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)

∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}

∧ trans Rel ∧ strong-reduction-bisimulation Rel (STCal Source Target))

(proof)

end
theory FullAbstraction
imports SourceTargetRelation
begin

9 Full Abstraction

An encoding is fully abstract w.r.t. some source term relation SRel and some target term relation
TRel if two source terms S1 and S2 form a pair (S1, S2) in SRel iff their literal translations form a
pair (enc S1, enc S2) in TRel.

abbreviation (in encoding) fully-abstract
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set ⇒ bool
  where
   fully-abstract SRel TRel ≡ ∀S1 S2. (S1, S2) ∈ SRel ←→ ([S1], [S2]) ∈ TRel

9.1 Trivial Full Abstraction Results

We start with some trivial full abstraction results. Each injective encoding is fully abstract w.r.t. to
the identity relation on the source and the identity relation on the target.

lemma (in encoding) inj-enc-is-fully-abstract-wrt-identities:
  assumes injectivity: ∀S1 S2. [S1] = [S2] → S1 = S2
  shows fully-abstract {(S1, S2). S1 = S2} {(T1, T2). T1 = T2}
⟨proof⟩

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

lemma (in encoding) fully-abstract-wrt-empty-relation:
  shows fully-abstract {} {}
⟨proof⟩

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

lemma (in encoding) fully-abstract-wrt-all-relation:
  shows fully-abstract {(S1, S2). True} {(T1, T2). True}
⟨proof⟩

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

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

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

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

lemma (in encoding) fully-abstract-wrt-trans-closure-of-source-relation:
  fixes RelS :: ('procS × 'procS) set
  assumes encRelS: ∀S1 S2. [S1] = [S2] → (S1, S2) ∈ RelS⁺
  shows ∃RelT. fully-abstract (RelS⁺) RelT
⟨proof⟩
For every encoding and every target term relation RelT there exists a source term relation RelS such that the encoding is fully abstract w.r.t. RelS and RelT.

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

- **Fixes** RelT :: (‘procT × ‘procT) set
- **Shows** ∃ RelS. fully-abstract RelS RelT

(\[proof\])

### 9.2 Fully Abstract Encodings

Thus, as long as we can choose one of the two relations, full abstraction is trivial. For fixed source and target term relations encodings are not trivially fully abstract. For all encodings and relations SRel and TRel we can construct a relation on the disjunctive union of source and target terms, whose reduction to source terms is SRel and whose reduction to target terms is TRel. But full abstraction ensures that each trans relation that relates source terms and their literal translations in both directions includes SRel iff it includes TRel restricted to translated source terms.

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

- **Fixes** Rel :: ((’procS, ’procT) Proc × (’procS, ’procT) Proc) set
- **And** SRel :: (’procS × ’procS) set
- **And** TRel :: (’procT × ’procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** sRel: SRel = { (S1, S2). SourceTerm S1, SourceTerm S2 ∈ Rel}
- **And** trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
- **shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel \(\leftrightarrow\) (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

(\[proof\])

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

- **Fixes** Rel :: ((’procS, ’procT) Proc × (’procS, ’procT) Proc) set
- **And** SRel :: (’procS × ’procS) set
- **And** TRel :: (’procT × ’procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** tRel: ∀ S1 S2. ([S1], [S2]) ∈ TRel \(\leftrightarrow\) (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, ’procT) Proc × (’procS, ’procT) Proc) set
- **And** SRel :: (’procS × ’procS) set
- **And** TRel :: (’procT × ’procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
- **Shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel \(\leftrightarrow\) (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

(\[proof\])

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

- **Fixes** Rel :: ((’procS, ’procT) Proc × (’procS, ’procT) Proc) set
- **And** SRel :: (’procS × ’procS) set
- **And** TRel :: (’procT × ’procT) set
- **Assumes** fullAbs: fully-abstract SRel TRel
- **And** encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
- **And** encL: ∀ S. (TargetTerm ([S]), SourceTerm S) ∈ Rel
- **And** trans: trans Rel
- **Shows** ∀ S1 S2. ([S1], [S2]) ∈ TRel \(\leftrightarrow\) (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 x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- shows refl SRel \iff (\forall S. ([S], [S]) \in TRel)

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

- fixes SRel : ('procS x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- surj: \forall T. \exists S. T = [S]
- shows refl SRel \iff refl TRel

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

- fixes SRel : ('procS x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- shows sym SRel \iff sym \{(T1, T2). \exists S1 S2. T1 = [S1] \land T2 = [S2] \land (T1, T2) \in TRel\}

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

- fixes SRel : ('procS x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- surj: \forall T. \exists S. T = [S]
- shows sym SRel \iff sym TRel

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

- fixes SRel : ('procS x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- shows trans SRel \iff trans \{(T1, T2). \exists S1 S2. T1 = [S1] \land T2 = [S2] \land (T1, T2) \in TRel\}

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

- fixes SRel : ('procS x 'procS) set
- and TRel : ('procT x 'procT) set
- assumes fullAbs: fully-abstract SRel TRel
- surj: \forall T. \exists S. T = [S]
- shows trans SRel \iff trans TRel

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 x 'procS) set
- and TRel : ('procT x 'procT) set
- and Pred : ('procS, 'procT) Proc \Rightarrow bool
- assumes fullAbs: fully-abstract SRel TRel
- and encP: enc-respects-pred Pred
- shows rel-preserve-pred \{(P, Q). \exists SP SQ. SP \in S P \land SQ \in S Q \land (SP, SQ) \in SRel\} Pred
  \iff rel-preserve-pred \{(P, Q). \exists SP SQ. [SP] \in T P \land [SQ] \in T Q \land ([SP], [SQ]) \in 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). ∃SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
≡ rel-preserves-binary-pred 
{(P, Q). ∃SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩
lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-binary-pred Pred
shows rel-reflects-pred {(P, Q). ∃SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
≡ rel-reflects-binary-pred 
{(P, Q). ∃SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
⟨proof⟩

9.3 Full Abstraction w.r.t. Preorders

If there however exists a trans relation Rel that relates source terms and their literal translations in both directions, then the encoding is fully abstract with respect to the reduction of Rel to source terms and the reduction of Rel to target terms.

lemma (in encoding) trans-source-target-relation-impl-full-abstraction:
If an encoding is fully abstract w.r.t. SRel and TRel, then we can conclude from a pair in indRelRTPO or indRelSTEQ on a pair in TRel and SRel.

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

- **Fixes** SRel :: (procS × procS) set
- **Assumes** enc: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ SRel ∧ (TargetTerm ([S]), SourceTerm S) ∈ SRel
- **And** trans: trans TRel
- **Shows** fully-abstract SRel TRel

(Proof)

If an encoding is fully abstract w.r.t. a preorder SRel on the source and a trans relation TRel on the target, then there exists a trans relation, namely indRelSTEQ, that relates source terms and their literal translations in both direction such that its reductions to source terms is SRel and its reduction to target terms is TRel.
lemma (in encoding) full-abstraction-wrt-preorders-impl-trans-source-target-relation:

defines \( SRel : (\text{proc}S \times \text{proc}S) \) set
and \( TRel : (\text{proc}T \times \text{proc}T) \) set
assumes fullAbs: fully-abstract \( SRel \) \( TRel \)
and reflS: ref\( t \) \( SRel \)
and transT: trans \( TRel \)

shows \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}
\wedge (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel})
\wedge \text{SRel} = \{(S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}
\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
\wedge \text{trans Rel} \)

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

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:

defines \( SRel : (\text{proc}S \times \text{proc}S) \) set
and \( TRel : (\text{proc}T \times \text{proc}T) \) set
shows (fully-abstract \( SRel \) \( TRel \) \wedge ref\( t \) \( SRel \) \wedge trans \( TRel \)) =
(\( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}
\wedge (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel})
\wedge \text{SRel} = \{(S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}
\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
\wedge \text{trans Rel} \)

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

9.4 Full Abstraction w.r.t. Equivalences

If there exists a relation Rel that relates source terms and their literal translations and whose sym closure is trans, then the encoding is fully abstract with respect to the reduction of the sym closure of Rel to source/target terms.

lemma (in encoding) source-target-relation-with-trans-symcl-impl-full-abstraction:

defines \( \text{Rel} : (\text{proc}S, \text{proc}T) \text{Proc} \times (\text{proc}S, \text{proc}T) \text{Proc} \) set
assumes enc: \( \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}
\wedge \text{trans: trans (symcl Rel)}
\wedge \text{symcl Rel} \)

shows fully-abstract \( \{(S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{symcl Rel}\} \)
\( \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{symcl Rel}\} \)

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

If an encoding is fully abstract w.r.t. the equivalences \( SRel \) \( TRel \), then there exists a preorder, namely \( \text{indRelRSTPO} \), 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:

defines \( SRel : (\text{proc}S \times \text{proc}S) \) set
and \( TRel : (\text{proc}T \times \text{proc}T) \) set
assumes fullAbs: fully-abstract \( SRel \) \( TRel \)
and reflT: ref\( t \) \( TRel \)
and symmT: sym \( TRel \)
and transT: trans \( TRel \)

shows \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}
\wedge \text{SRel} = \{(S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{symcl Rel}\}
\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{symcl Rel}\}
\wedge \text{preorder (symcl Rel)} \)

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

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

defines \( SRel : (\text{proc}S \times \text{proc}S) \) set
and \( TRel : (\text{proc}T \times \text{proc}T) \) set
assumes fullAbs: fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+)
s-shows \exists 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 (symcl Rel)}

(proof)

\textbf{lemma (in encoding)} fully-abstract-wrt-preorders-impl-source-target-relation-is-trans:
\text{fixes SRel : ("procS x "procS) set}
\text{and TRel : ("procT x "procT) set}
\text{assumes fullAbs: fully-abstract SRel TRel}
\text{shows \exists 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{refl SRel \land trans TRel})
\iff trans (Rel \cup \{(P, Q), S \in T \land S \in S Q\}))

(proof)

\textbf{lemma (in encoding)} fully-abstract-wrt-preorders-impl-source-target-relation-is-trans-B:
\text{fixes SRel : ("procS x "procS) set}
\text{and TRel : ("procT x "procT) set}
\text{assumes fullAbs: fully-abstract SRel TRel}
\text{and reflT: refl TRel}
\text{and transT: trans TRel}
\text{shows \exists 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 (Rel \cup \{(P, Q), 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 if 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.

\textbf{lemma (in encoding)} fully-abstract-wrt-equivalences-iff-symcl-source-target-relation-is-preorder:
\text{fixes SRel : ("procS x "procS) set}
\text{and TRel : ("procT x "procT) set}
\text{shows (fully-abstract SRel TRel \land \text{equivalence TRel}) =}
\exists 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 (symcl Rel)}

(proof)

\textbf{lemma (in encoding)} fully-abstract-iff-symcl-source-target-relation-is-preorder:
\text{fixes SRel : ("procS x "procS) set}
\text{and TRel : ("procT x "procT) set}
\text{shows fully-abstract ((symcl (SRel=))+) ((symcl (TRel=))+) =}
\exists 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 (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.

\textbf{lemma (in encoding)} full-abstraction-impl-indRelSTEQ-is-trans:
Whenever an encoding induces a trans relation that includes SRel and TRel and relates source terms to their literal translations in both directions, the encoding is fully abstract w.r.t. SRel and TRel.

**Lemma (in encoding) trans-source-target-relation-impl-fully-abstract:**

- **Fixes** $Rel ::= (\langle procS \times procT \rangle \text{ Proc} \times (\langle procS, procT \rangle \text{ Proc})$ set
- **Assumes** $\text{fullAbs}$: fully-abstract $SRel \ TRel$
  - $\text{rel}$: $Rel = ((\text{indRelSTEQ} SRel \ TRel) \cup \{(P, Q). (P \in ProcS \land Q \in ProcS) \lor (P \in ProcT \land Q \in ProcT)\} \cup \{(P, Q). \exists SP SQ. SP \in S \land SQ \in Q \land (SP, SQ) \in SRel\}) \lor \{(TP, TQ). TP \in T \land TQ \in T \land (TP, TQ) \in TRel\})$
  - $\text{shows}$ $(\text{refl} SRel \land \text{trans} TRel) = \text{trans} Rel$
    - (proof)

Assume TRel is a preorder. Then an encoding is fully abstract w.r.t. SRel and TRel if there exists a relation that relates at least all source terms to their literal translations, includes SRel and TRel, and whose union with the relation that relates exactly all literal translations to their source terms is trans.

**Lemma (in encoding) source-target-relation-with-trans-impl-full-abstraction:**

- **Fixes** $Rel ::= (\langle procS, procT \rangle \text{ Proc} \times (\langle procS, procT \rangle \text{ Proc})$ set
- **Assumes** $\text{enc}$: $\forall S. \exists T. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in Rel$ and $SRel = ((S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in Rel}$
  - $\text{shows}$ $\{\{S1, S2\}, (\text{TargetTerm} T1, \text{TargetTerm} T2) \in Rel\}$
    - (proof)

**Lemma (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-is-transB:**

- **Fixes** $SRel ::= (\langle procS \times procS \rangle$ set
- **Assumes** $\text{procord}$: preorder $TRel$
  - $\text{shows}$ fully-abstract $SRel \ TRel$
    - (proof)

The same holds if to obtain transitivity the union may contain additional pairs that do neither relate two source nor two target terms.

**Lemma (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-union-is-trans:**

- **Fixes** $SRel ::= (\langle procS \times procS \rangle$ set
- **Assumes** $\text{procord}$: preorder $TRel$
  - $\text{shows}$ $(\text{fully-abstract} SRel \ TRel \land \text{refl} SRel \land \text{trans} TRel) =$
    - (proof)
10 Combining Criteria

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

\textbf{lemma (in encoding) criterion-iff-source-target-relation-impl-indRelR:}
\begin{proof}
\begin{align*}
&\text{fixes Cond :: } (\text{proc} S \Rightarrow \text{proc} T) \Rightarrow \text{bool} \\
&\text{and Pred :: } (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times \langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) \text{ set } \Rightarrow \text{bool} \\
&\text{assumes Cond enc } = (3 \text{ Rel} , (\forall S \cdot (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \land \text{Pred Rel}) \\
&\text{shows Cond enc } = (3 \text{ Rel}', \text{Pred } (\text{indRelR } \cup \text{Rel}'))
\end{align*}
\end{proof}

\textbf{lemma (in encoding) combine-conditions-on-pairs-of-relations:}
\begin{proof}
\begin{align*}
&\text{fixes RelA RelB } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) \text{ set} \\
&\text{and CondA CondB } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) ) \Rightarrow \text{bool} \\
&\text{assumes } (\forall (P, Q) \in \text{RelA} \cdot \text{CondA } (P, Q) \\
&\text{and } (\forall (P, Q) \in \text{RelB} \cdot \text{CondB } (P, Q) \\
&\text{shows } (\forall (P, Q) \in \text{RelA } \cap \text{RelB} \cdot \text{CondA } (P, Q)) \land (\forall (P, Q) \in \text{RelA } \cap \text{RelB} \cdot \text{CondB } (P, Q))
\end{align*}
\end{proof}

\textbf{lemma (in encoding) combine-conditions-on-sets-of-relations:}
\begin{proof}
\begin{align*}
&\text{fixes RelA RelB } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) \text{ set} \\
&\text{and Cond } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) ) \Rightarrow \text{bool} \\
&\text{assumes } (\forall (P, Q) \in \text{RelA} \cdot \text{Cond } (P, Q) \\
&\text{and } \text{Cond Rel } \land \text{Rel } \subseteq \text{RelA} \\
&\text{shows } \text{Cond Rel } \land (\forall (P, Q) \in \text{Rel} \cdot \text{CondA } (P, Q))
\end{align*}
\end{proof}

\textbf{lemma (in encoding) combine-conditions-on-sets-and-pairs-of-relations:}
\begin{proof}
\begin{align*}
&\text{fixes RelA RelB } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) \text{ set} \\
&\text{and Cond } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) ) \Rightarrow \text{bool} \\
&\text{assumes } (\forall (P, Q) \in \text{RelA} \cdot \text{Cond } (P, Q) \\
&\text{and } (\forall (P, Q) \in \text{RelB} \cdot \text{CondB } (P, Q) \\
&\text{and } \text{Cond Rel } \land \text{Rel } \subseteq \text{RelA } \land \text{Rel } \subseteq \text{RelB} \\
&\text{shows } \text{Cond Rel } \land (\forall (P, Q) \in \text{Rel} \cdot \text{CondA } (P, Q)) \land (\forall (P, Q) \in \text{Rel} \cdot \text{CondB } (P, Q))
\end{align*}
\end{proof}

We mapped several criteria on conditions on relations that relate at least all source terms and their
literal translations. The following lemmata help us to combine such conditions by switching to the
witness indRelR.

\textbf{lemma (in encoding) combine-conditions-on-relations-indRelR:}
\begin{proof}
\begin{align*}
&\text{fixes RelA RelB } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) \text{ set} \\
&\text{and Cond } :: (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc } \times (\langle \text{proc} S, \text{proc} T \rangle \text{ Proc}) ) \Rightarrow \text{bool} \\
&\text{assumes } A1: \forall S \cdot (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{RelA} \\
&\text{and } A2: \forall (P, Q) \in \text{RelA} \cdot \text{CondA } (P, Q)
\end{align*}
\end{proof}
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 $\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{CondA} (P, Q))$

$s:\ R_{\text{div}} \bigg( (\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{CondA} (P, Q)) \wedge (\forall (P, Q) \in \text{Rel}. \text{CondB} (P, Q)) \wedge \text{Cond Rel})$

$\langle \text{proof} \rangle$

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

fixes $\text{PredA, PredB} :: \langle \text{procS, 'procT} \text{Proc} \Rightarrow \text{bool} \rangle$

shows $(\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \wedge \text{rel-respects-pred Rel PredA})$

$s: \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}$

$\langle \text{proof} \rangle$

10.1 Divergence Reflection and Success Sensitiveness

We combine results on divergence reflection and success sensitiveness to analyse their combined effect on an encoding function. An encoding is success sensitive and reflects divergence iff there exists a relation that relates source terms and their literal translations that reflects divergence and respects success.

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

fixes $\text{success :: 'barb}$

shows $(\text{enc-weakly-respects-barb-set \{success\} \wedge \text{enc-reflects-divergence}})$

$s: \text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}$

$\langle \text{proof} \rangle$

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

fixes $\text{success :: 'barb}$

shows $(\text{enc-respects-barb-set \{success\} \wedge \text{enc-reflects-divergence}})$

$s: \text{rel-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}$

$\langle \text{proof} \rangle$

10.2 Adding Operational Correspondence

The effect of operational correspondence includes conditions (TRel is included, transitivity) that require a witness like indRelRTPO. In order to combine operational correspondence with success sensitiveness, we show that if the encoding and TRel (weakly) respects bars than indRelRTPO (weakly) respects bars. Since success is only a specific kind of bars, the same holds for success sensitiveness.

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

fixes $\text{success :: 'barb}$

and TRel $:: \langle \text{procT} \times \text{procT} \rangle$ set

assumes encRS: $\text{enc-weakly-respects-barb-set \{success\}}$

and trelPS: $\text{rel-weakly-preserves-barb-set TRel TWB \{success\}}$

and trelRS: $\text{rel-weakly-reflects-barb-set TRel TWB \{success\}}$

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

$\langle \text{proof} \rangle$

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

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

assumes encRS: $\text{enc-weakly-respects-bars}$

and trelPS: $\text{rel-weakly-preserves-bars TRel TWB}$

and trelRS: $\text{rel-weakly-reflects-bars TRel TWB}$

shows $\text{rel-weakly-respects-bars \{indRelRTPO TRel\} (STCalWB SWB TWB)}$
lemma (in encoding-wrt-barbs) enc-and-TRel-impl-indRelRTPO-respects-success:

\[ \text{fixes } \text{'barbs}, \text{'procT} \times \text{'procT} \text{ set} \]
\[ \text{and } \text{TRel }:: \langle \text{procT} \times \text{procT} \rangle \text{ set} \]
\[ \text{assumes } \text{encRS: enc-respects-barb-set } \{ \text{success} \} \]
\[ \text{and } \text{trePS: rel-preserves-barb-set } \text{TRel } \text{ TWB } \{ \text{success} \} \]
\[ \text{and } \text{treRS: rel-reflects-barb-set } \text{TRel } \text{ TWB } \{ \text{success} \} \]
\[ \text{shows } \text{rel-respects-barb-set } (\text{indRelRTPO } \text{TRel}) (\text{STCalWB SWB TWB}) \{ \text{success} \} \]

-proof-

lemma (in encoding-wrt-barbs) enc-and-TRel-impl-indRelRTPO-respects-barbs:

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

-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:

\[ \text{fixes } \text{'barbs}, \text{'procT} \times \text{'procT} \text{ set} \]
\[ \text{and } \text{TRel }:: \langle \text{procT} \times \text{procT} \rangle \text{ set} \]
\[ \text{shows } \text{(operational-corresponding } (\text{TRel}^*) \]
\[ \land \text{ weak-reduction-bisimulation } (\text{TRel}^*) \text{ Target} \]
\[ \land \text{ enc-weakly-respects-barb-set } \{ \text{success} \} \]
\[ \land \text{ rel-weakly-respects-barb-set } \text{TRel } \text{ TWB } \{ \text{success} \} \]
\[ = (\exists \text{Rel. } (\forall S. (\text{SourceTerm S, TargetTerm } (\text{[S]}))) \in \text{Rel}) \]
\[ \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel } \rightarrow (\text{TargetTerm T_1, TargetTerm T_2}) \in \text{Rel}) \]
\[ \land (\forall T_1 T_2. (\text{TargetTerm T_1, TargetTerm T_2}) \in \text{Rel } \rightarrow (T_1, T_2) \in \text{TRel}^+) \]
\[ \land (\forall S T. (\text{SourceTerm S, TargetTerm T}) \in \text{Rel } \rightarrow (\text{[S], T}) \in \text{TRel}^+) \]
\[ \land \text{ weak-reduction-bisimulation Rel } (\text{STCal Source Target}) \]
\[ \land \text{ rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \{ \text{success} \} \]

-proof-

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

\[ \text{fixes } \text{'barbs}, \text{'procT} \times \text{'procT} \text{ set} \]
\[ \text{and } \text{TRel }:: \langle \text{procT} \times \text{procT} \rangle \text{ set} \]
\[ \text{shows } \text{(operational-corresponding } (\text{TRel}^*) \]
\[ \land \text{ weak-reduction-bisimulation } (\text{TRel}^*) \text{ Target} \]
\[ \land \text{ enc-weakly-respects-barbs } \land \text{ enc-weakly-respects-barb-set } \{ \text{success} \} \]
\[ \land \text{ rel-weakly-respects-barbs } \text{TRel } \text{ TWB } \land \text{ rel-weakly-respects-barb-set } \text{TRel } \text{ TWB } \{ \text{success} \} \]
\[ = (\exists \text{Rel. } (\forall S. (\text{SourceTerm S, TargetTerm } (\text{[S]}))) \in \text{Rel}) \]
\[ \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel } \rightarrow (\text{TargetTerm T_1, TargetTerm T_2}) \in \text{Rel}) \]
\[ \land (\forall T_1 T_2. (\text{TargetTerm T_1, TargetTerm T_2}) \in \text{Rel } \rightarrow (T_1, T_2) \in \text{TRel}^+) \]
\[ \land (\forall S T. (\text{SourceTerm S, TargetTerm T}) \in \text{Rel } \rightarrow (\text{[S], T}) \in \text{TRel}^+) \]
\[ \land \text{ weak-reduction-bisimulation Rel } (\text{STCal Source Target}) \]
\[ \land \text{ rel-weakly-respects-barbs Rel } (\text{STCalWB SWB TWB}) \]
\[ \land \text{ rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TWB}) \{ \text{success} \} \]

-proof-

lemma (in encoding-wrt-barbs) OC-SS-wpreorder-iff-source-target-rel:

\[ \text{fixes } \text{'barbs}, \text{'procT} \times \text{'procT} \text{ set} \]
\[ \text{and } \text{TRel }:: \langle \text{procT} \times \text{procT} \rangle \text{ set} \]
\[ \text{shows } \text{(operational-corresponding } \text{TRel }\land \text{ preorder } \text{TRel}\land \text{ weak-reduction-bisimulation } \text{TRel } \text{ Target} \]
\[ \land \text{ enc-weakly-respects-barb-set } \{ \text{success} \} \]
\[ \land \text{ rel-weakly-respects-barb-set } \text{TRel } \text{ TWB } \{ \text{success} \} \]
\[ = (\exists \text{Rel. } (\forall S. (\text{SourceTerm S, TargetTerm } (\text{[S]}))) \in \text{Rel}) \]

79
\[\begin{align*}
\land TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2) \in Rel\} \\
\land (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel) \\
\land weak-reduction-bisimulation Rel (STCal Source Target) \land preorder Rel \\
\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\} 
\end{align*}\]

(\textit{proof})

\textbf{lemma (in encoding-wrt-barbs)} \textit{OC-SS-RB-wrt-preorder-iff-source-target-rel:}

\begin{itemize}
\item \textbf{fixes} success :: 'bars
\item \textbf{and} TRel :: ('procT × 'procT) set
\item \textbf{shows} (operational-corresponding TRel \land preorder TRel \land weak-reduction-bisimulation TRel Target \\
\land enc-weakly-respects-barbs \land rel-weakly-respects-barbs TRel TWB \\
\land enc-weakly-respects-barb-set \{success\} \\
\land rel-weakly-respects-barb-set TRel TWB \{success\}) \\
= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
\land TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2) \in Rel\} \\
\land (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel) \\
\land weak-reduction-bisimulation Rel (STCal Source Target) \land preorder Rel \\
\land rel-weakly-respects-barbs Rel (STCalWB SWB TWB) \\
\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}) 
\end{itemize}

(\textit{proof})

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

\textbf{lemma (in encoding-wrt-barbs)} \textit{WOC-SS-wrt-preorder-iff-source-target-rel:}

\begin{itemize}
\item \textbf{fixes} success :: 'bars
\item \textbf{and} TRel :: ('procT × 'procT) set
\item \textbf{shows} (weakly-operational-corresponding TRel \land preorder TRel \\
\land weak-reduction-correspondence-simulation TRel Target \\
\land enc-weakly-respects-barb-set \{success\} \\
\land rel-weakly-respects-barb-set TRel TWB \{success\}) \\
= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
\land TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2) \in Rel\} \\
\land (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel) \\
\land weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel \\
\land rel-weakly-respects-barbs Rel (STCalWB SWB TWB) \\
\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}) 
\end{itemize}

(\textit{proof})

\textbf{lemma (in encoding-wrt-barbs)} \textit{SOC-SS-RB-wrt-preorder-iff-source-target-rel:}

\begin{itemize}
\item \textbf{fixes} success :: 'bars
\item \textbf{and} TRel :: ('procT × 'procT) set
\item \textbf{shows} (weakly-operational-corresponding TRel \land preorder TRel \\
\land weak-reduction-correspondence-simulation TRel Target \\
\land enc-weakly-respects-barbs \land enc-weakly-respects-barb-set \{success\} \\
\land rel-weakly-respects-barbs TRel TWB \land rel-weakly-respects-barb-set TRel TWB \{success\}) \\
= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel) \\
\land TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2) \in Rel\} \\
\land (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel) \\
\land weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel \\
\land rel-weakly-respects-barbs Rel (STCalWB SWB TWB) \\
\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}) 
\end{itemize}

(\textit{proof})

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

\textbf{lemma (in encoding-wrt-barbs)} \textit{SOC-SS-wrt-preorder-iff-source-target-rel:}

\begin{itemize}
\item \textbf{fixes} success :: 'bars
\end{itemize}
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}`)
  = (∀ 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**) enc-and-TRelimpl-indRelRTPO-reflect-divergence:

- **Fixes** `TRel :: ('procT × 'procT) set`
- **Assumes** encRD: enc-reflects-divergence
- **And** trelRD: rel-reflects-divergence `TRel Target`
- **Shows** rel-reflects-divergence (indRelRTPO TRel) (STCal Source Target)

**(proof)**

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

- **Fixes** `success :: 'barbs`
- **And** `TRel :: ('procT × 'procT) set`
- **Shows** (operational-coringresponding `(TRel+)`)
  ∧ weak-reduction-bisimulation `(TRel+) Target`
  ∧ enc-weakly-respects-barb-set `{success}`
  ∧ rel-weakly-respects-barb-set `TRel TWB {success}`
  = (∀ 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-coringresponding `TRel ∧ preorder TRel`
  ∧ weak-reduction-correspondence-simulation `TRel Target`
  ∧ enc-weakly-respects-barb-set `{success}`
  ∧ rel-weakly-respects-barb-set `TRel TWB {success}`
  ∧ enc-reflects-divergence ∧ rel-reflects-divergence `TRel Target`
  = (∀ S . (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2) . (TargetTerm T1, TargetTerm T2) ∈ Rel}
\[(\forall S, T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})\]
\[(\forall S, T. \text{rel-weakly-respects-barb-set Rel} (\text{STCal Source Target}) \wedge \text{preorder Rel})\]
\[
\text{fully abstract w.r.t. SRel and TRel and operationally corresponding w.r.t. TRel such that TRel is a}\]
\[
\text{transitive bisimulation.}
\]
\[
\text{To combine full abstraction and operational correspondence we consider a symmetric version of the}\]
\[
\text{induced relation and assume that the relations SRel and TRel are equivalences. Then an encoding is}\]
\[
\text{fully abstract w.r.t. SRel and TRel and operationally corresponding w.r.t. TRel such that TRel is a}\]
\[
\text{bisimulation iff the induced relation contains both SRel and TRel and is a transitive bisimulation.}
\]
\[
\text{lemma (in encoding) OC-SS-DR-wrt-preorder-iff-source-target-rel:}\]
\[
\text{fixes success :: 'barbs}\]
\[
\text{and TRel :: ('procT × 'procT) set}\]
\[
\text{shows (strongly-operational-corresponding TRel} \wedge \text{preorder TRel} \wedge \text{weak-reduction-bisimulation TRel Target}\]
\[
\wedge \text{enc-respects-barb-set} \{\text{success}\} \wedge \text{rel-respects-barb-set TRel TWB} \{\text{success}\}\]
\[
\wedge \text{enc-reflects-divergence} \wedge \text{rel-reflects-divergence TRel Target}\]
\[
\Rightarrow (\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})\]
\[
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}\]
\[
\wedge (\forall S, T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})\]
\[
\wedge \text{weak-reduction-bisimulation Rel} (\text{STCal Source Target}) \wedge \text{preorder Rel}\]
\[
\wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB)} \{\text{success}\}\]
\[
\wedge \text{rel-reflects-divergence Rel} (\text{STCal Source Target})\]
\[
\text{(proof)}
\]
\[
\text{lemma (in encoding) SOC-SS-DR-wrt-preorder-iff-source-target-rel:}\]
\[
\text{fixes success :: 'barbs}\]
\[
\text{and TRel :: ('procT × 'procT) set}\]
\[
\text{shows (strongly-operational-corresponding TRel} \wedge \text{preorder TRel}\]
\[
\wedge \text{enc-respects-barb-set} \{\text{success}\} \wedge \text{rel-respects-barb-set TRel TWB} \{\text{success}\}\]
\[
\wedge \text{enc-reflects-divergence} \wedge \text{rel-reflects-divergence TRel Target}\]
\[
\Rightarrow (\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})\]
\[
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}\]
\[
\wedge (\forall S, T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})\]
\[
\wedge \text{weak-reduction-bisimulation Rel} (\text{STCal Source Target}) \wedge \text{preorder Rel}\]
\[
\wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB)} \{\text{success}\}\]
\[
\wedge \text{rel-reflects-divergence Rel} (\text{STCal Source Target})\]
\[
\text{(proof)}
\]
\[\text{10.3 Full Abstraction and Operational Correspondence}\]

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.

\[
\text{lemma (in encoding) FS-OC-modulo-equivalences-iff-source-target-relation:}\]
\[
\text{fixes SRel :: ('procS × 'procS) set}\]
\[
\text{and TRel :: ('procT × 'procT) set}\]
\[
\text{assumes eqS: equivalence SRel}\]
\[
\text{and eqT: equivalence TRel}\]
\[
\text{shows fully-abstract SRel TRel}\]
\[
\wedge \text{operational-corresponding TRel} \wedge \text{weak-reduction-bisimulation TRel Target}\]
\[
\leftarrow (\exists \text{Rel}.\]
\[
(\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \wedge (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel})\]
\[
\wedge \text{SRel} = \{(S1, S2), (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}\]
\[
\wedge \text{TRel} = \{(T1, T2), (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}\]
\[
\wedge \text{trans Rel} \wedge \text{weak-reduction-bisimulation Rel} (\text{STCal Source Target})\]
\[
\text{(proof)}
\]
\[
\text{lemma (in encoding) FA-SOC-modulo-equivalences-iff-source-target-relation:}\]
\[
\text{fixes SRel :: ('procS × 'procS) set}\]
\[
\text{and TRel :: ('procT × 'procT) set}\]
\[
\text{assumes eqS: equivalence SRel}\]
and \( eqT \): equivalence \( T\text{Rel} \)

shows fully-abstract \( S\text{Rel} \land \text{strongly-operational-corresponding } T\text{Rel} \)
\( \land \text{strong-reduction-bisimulation } T\text{Rel } \forall Target \Leftarrow (\exists Rel. \)
\( (\forall S. (Source\text{Term } S, Target\text{Term } ([S])) \in Rel \land (Target\text{Term } ([S]), Source\text{Term } S) \in Rel) \)
\( \land \ S\text{Rel} = \{(S1, S2). (Source\text{Term } S1, Source\text{Term } S2) \in Rel\} \)
\( \land \ T\text{Rel} = \{(T1, T2). (Target\text{Term } T1, Target\text{Term } T2) \in Rel\} \land \text{trans Rel} \land \text{strong-reduction-bisimulation Rel } (STCal Source Target) \)

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

An encoding that is fully abstract w.r.t. the equivalences \( S\text{Rel} \) and \( T\text{Rel} \) and operationally corresponding w.r.t. \( T\text{Rel} \) ensures that \( S\text{Rel} \) is a bisimulation iff \( T\text{Rel} \) is a bisimulation.

lemma (in encoding) \text{FA-and-OC-and-TRel-impl-SRel-bisimulation}:
fixes \( S\text{Rel} : (\text{procS } \times \text{procS}) \) set
and \( T\text{Rel} : (\text{procT } \times \text{procT}) \) set
assumes fullAbs: fully-abstract \( S\text{Rel} \land \text{TRel} \)
and opCom: operational-complete \( T\text{Rel} \)
and opSou: operational-sound \( T\text{Rel} \)
and symmT: sym \( T\text{Rel} \)
and transT: trans \( T\text{Rel} \)
and bisimT: weak-reduction-bisimulation \( T\text{Rel } Target \)

shows weak-reduction-bisimulation \( S\text{Rel } Source \)

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

lemma (in encoding) \text{FA-and-SOC-and-TRel-impl-SRel-strong-bisimulation}:
fixes \( S\text{Rel} : (\text{procS } \times \text{procS}) \) set
and \( T\text{Rel} : (\text{procT } \times \text{procT}) \) set
assumes fullAbs: fully-abstract \( S\text{Rel} \land \text{TRel} \)
and opCom: strongly-operational-complete \( T\text{Rel} \)
and opSou: strongly-operational-sound \( T\text{Rel} \)
and symmT: sym \( T\text{Rel} \)
and transT: trans \( T\text{Rel} \)
and bisimT: strong-reduction-bisimulation \( T\text{Rel } Target \)

shows strong-reduction-bisimulation \( S\text{Rel } Source \)

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

lemma (in encoding) \text{FA-and-OC-impl-SRel-iff-TRel-bisimulation}:
fixes \( S\text{Rel} : (\text{procS } \times \text{procS}) \) set
and \( T\text{Rel} : (\text{procT } \times \text{procT}) \) set
assumes fullAbs: fully-abstract \( S\text{Rel} \land \text{TRel} \)
and opCor: operational-corresponding \( T\text{Rel} \)
and symmT: sym \( T\text{Rel} \)
and transT: trans \( T\text{Rel} \)
and sury: \( \forall T. \exists S. T = [S] \)

shows weak-reduction-bisimulation \( S\text{Rel } Source \leftrightarrow \text{weak-reduction-bisimulation } T\text{Rel } Target \)

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

end