

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

```

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

```

1 Relations

1.1 Basic Conditions

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

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

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

A symmetric preorder is an equivalence.

lemma *symm-preorder-is-equivalence:*

fixes *Rel* :: ('a × 'a) set

assumes *preorder Rel*

and *sym Rel*

shows *equivalence Rel*

<proof>

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

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

symcl Rel = *Rel* \cup *Rel*⁻¹

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

lemma *elem-of-symcl:*

fixes *Rel* :: ('a × 'a) set

and *a b* :: 'a

assumes *elem: (a, b) ∈ Rel*

shows *(a, b) ∈ symcl Rel*

and *(b, a) ∈ symcl Rel*

<proof>

The symmetric closure of a relation is symmetric.

lemma *sym-symcl:*

fixes *Rel* :: ('a × 'a) set

shows *sym (symcl Rel)*

<proof>

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

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

fixes *Rel* :: ('a × 'a) set

shows *symcl (Rel⁼) = (symcl Rel)⁼*

<proof>

The symmetric closure of a reflexive relation is reflexive.

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

fixes *Rel* :: ('a × 'a) set

and *A* :: 'a set

assumes *refl-on A Rel*

shows *refl-on A (symcl Rel)*

<proof>

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

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

fixes $Rel :: ('a \times 'a) \text{ set}$
shows $(\text{symcl } (Rel^=))^+ = (\text{symcl } Rel)^*$
 $\langle \text{proof} \rangle$

The reflexive closure of a symmetric relation is symmetric.

lemma *sym-reflcl-of-symm-rel*:
fixes $Rel :: ('a \times 'a) \text{ set}$
assumes $\text{sym } Rel$
shows $\text{sym } (Rel^=)$
 $\langle \text{proof} \rangle$

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

lemma *reflcl-of-refl-rel*:
fixes $Rel :: ('a \times 'a) \text{ set}$
assumes $\text{refl } Rel$
shows $Rel^= = Rel$
 $\langle \text{proof} \rangle$

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

lemma *symm-closure-of-symm-rel*:
fixes $Rel :: ('a \times 'a) \text{ set}$
assumes $\text{sym } Rel$
shows $\text{symcl } Rel = Rel$
 $\langle \text{proof} \rangle$

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

lemma *rtrancl-of-preorder*:
fixes $Rel :: ('a \times 'a) \text{ set}$
assumes $\text{preorder } Rel$
shows $Rel^* = Rel$
 $\langle \text{proof} \rangle$

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

lemma *refl-trans-closure-subset-of-refl-symm-trans-closure*:
fixes $Rel :: ('a \times 'a) \text{ set}$
shows $Rel^* \subseteq (\text{symcl } (Rel^=))^+$
 $\langle \text{proof} \rangle$

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

lemma *symm-closure-of-preorder-is-trans*:
fixes $Rel :: ('a \times 'a) \text{ set}$
assumes $\text{condA}: \forall a b c. (a, b) \in Rel \wedge (c, b) \in Rel \wedge (a, c) \notin Rel$
 $\quad \longrightarrow (b, a) \in Rel \vee (b, c) \in Rel$
and $\text{condB}: \forall a b c. (a, b) \in Rel \wedge (a, c) \in Rel \wedge (b, c) \notin Rel$
 $\quad \longrightarrow (b, a) \in Rel \vee (c, a) \in Rel$
and $\text{reflR}: \text{refl } Rel$
and $\text{tranR}: \text{trans } Rel$
shows $\text{trans } (\text{symcl } Rel)$
 $\langle \text{proof} \rangle$

1.2 Preservation, Reflection, and Respection of Predicates

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

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

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

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

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

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

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

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

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

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

lemma *symm-relation-impl-preservation-equals-reflection*:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ bool
assumes *symm*: sym Rel
shows *rel-preserves-pred* Rel Pred = *rel-reflects-pred* Rel Pred
and *rel-preserves-pred* Rel Pred = *rel-respects-pred* Rel Pred
and *rel-reflects-pred* Rel Pred = *rel-respects-pred* Rel Pred
⟨proof⟩

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

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

lemma *preservation-and-closures*:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ bool
assumes *preservation*: *rel-preserves-pred* Rel Pred
shows *rel-preserves-pred* (Rel⁼) Pred
and *rel-preserves-pred* (Rel⁺) Pred
and *rel-preserves-pred* (Rel^{*}) Pred
⟨proof⟩

lemma *preservation-of-binary-predicates-and-closures*:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ 'b ⇒ bool
assumes *preservation*: *rel-preserves-binary-pred* Rel Pred
shows *rel-preserves-binary-pred* (Rel⁼) Pred
and *rel-preserves-binary-pred* (Rel⁺) Pred
and *rel-preserves-binary-pred* (Rel^{*}) Pred
⟨proof⟩

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

lemma *reflection-and-closures*:

```

fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ bool
assumes reflection: rel-reflects-pred Rel Pred
shows rel-reflects-pred (Rel=) Pred
and rel-reflects-pred (Rel+) Pred
and rel-reflects-pred (Rel*) Pred
⟨proof⟩

```

```

lemma reflection-of-binary-predicates-and-closures:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ 'b ⇒ bool
assumes reflection: rel-reflects-binary-pred Rel Pred
shows rel-reflects-binary-pred (Rel=) Pred
and rel-reflects-binary-pred (Rel+) Pred
and rel-reflects-binary-pred (Rel*) Pred
⟨proof⟩

```

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

```

lemma respection-and-closures:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ bool
assumes respection: rel-respects-pred Rel Pred
shows rel-respects-pred (Rel=) Pred
and rel-respects-pred (symcl Rel) Pred
and rel-respects-pred (Rel+) Pred
and rel-respects-pred (symcl (Rel=)) Pred
and rel-respects-pred (Rel*) Pred
and rel-respects-pred ((symcl (Rel=))+) Pred
⟨proof⟩

```

```

lemma respection-of-binary-predicates-and-closures:
fixes Rel :: ('a × 'a) set
and Pred :: 'a ⇒ 'b ⇒ bool
assumes respection: rel-respects-binary-pred Rel Pred
shows rel-respects-binary-pred (Rel=) Pred
and rel-respects-binary-pred (symcl Rel) Pred
and rel-respects-binary-pred (Rel+) Pred
and rel-respects-binary-pred (symcl (Rel=)) Pred
and rel-respects-binary-pred (Rel*) Pred
and rel-respects-binary-pred ((symcl (Rel=))+) Pred
⟨proof⟩

```

```

end
theory ProcessCalculi
imports Relations
begin

```

2 Process Calculi

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

2.1 Reduction Semantics

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

```

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

```

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

abbreviation $step :: 'proc \Rightarrow 'proc \text{ processCalculus} \Rightarrow 'proc \Rightarrow bool$
 $(- \mapsto - - [70, 70, 70] 80)$
where
 $P \mapsto Cal Q \equiv Reductions Cal P Q$

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

primrec $nSteps$
 $:: 'proc \Rightarrow 'proc \text{ processCalculus} \Rightarrow nat \Rightarrow 'proc \Rightarrow bool$
 $(- \mapsto - - [70, 70, 70, 70] 80)$
where
 $P \mapsto Cal^0 Q = (P = Q) |$
 $P \mapsto Cal^{Suc\ n} Q = (\exists P'. P \mapsto Cal^n P' \wedge P' \mapsto Cal Q)$

definition $steps$
 $:: 'proc \Rightarrow 'proc \text{ processCalculus} \Rightarrow 'proc \Rightarrow bool$
 $(- \mapsto * - [70, 70, 70] 80)$
where
 $P \mapsto Cal* Q \equiv \exists n. P \mapsto Cal^n Q$

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

definition $divergent$
 $:: 'proc \Rightarrow 'proc \text{ processCalculus} \Rightarrow bool$
 $(- \mapsto \omega [70, 70] 80)$
where
 $P \mapsto (Cal)\omega \equiv \forall P'. P \mapsto Cal* P' \longrightarrow (\exists P''. P' \mapsto Cal P'')$

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

lemma $steps-refl$:
fixes $Cal :: 'proc \text{ processCalculus}$
and $P :: 'proc$
shows $P \mapsto Cal* P$
 $\langle proof \rangle$

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

lemma $step-to-steps$:
fixes $Cal :: 'proc \text{ processCalculus}$
and $P P' :: 'proc$
assumes $step: P \mapsto Cal P'$
shows $P \mapsto Cal* P'$
 $\langle proof \rangle$

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 $Cal :: 'proc \text{ processCalculus}$
and $n1\ n2 :: nat$
shows $\forall P\ Q\ R. P \mapsto Cal^{n1} Q \wedge Q \mapsto Cal^{n2} R \longrightarrow P \mapsto Cal^{(n1 + n2)} R$
 $\langle proof \rangle$

lemma $steps-add$:
fixes $Cal :: 'proc \text{ processCalculus}$
and $P\ Q\ R :: 'proc$
assumes $A1: P \mapsto Cal* Q$
and $A2: Q \mapsto Cal* R$
shows $P \mapsto Cal* R$
 $\langle proof \rangle$

2.1.1 Observables or Barbs

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

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

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

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

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

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

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

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

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

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

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

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

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

A relation respects barbs if it preserves and reflects barbs.

abbreviation *rel-respects-barb-set*

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

abbreviation *rel-respects-barbs*

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

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

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

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

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

fixes $Rel :: ('proc \times 'proc) set$
and $CWB :: ('proc, 'barbs) calculusWithBarbs$
assumes *preservation: rel-preserves-barbs Rel CWB*
shows $rel-preserves-barbs (Rel^=) CWB$
and $rel-preserves-barbs (Rel^+) CWB$
and $rel-preserves-barbs (Rel^*) CWB$
 $\langle proof \rangle$

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

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

fixes $Rel :: ('proc \times 'proc) set$
and $CWB :: ('proc, 'barbs) calculusWithBarbs$
assumes *reflection: rel-reflects-barbs Rel CWB*
shows $rel-reflects-barbs (Rel^=) CWB$
and $rel-reflects-barbs (Rel^+) CWB$
and $rel-reflects-barbs (Rel^*) CWB$
 $\langle proof \rangle$

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

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

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

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

abbreviation *rel-weakly-preserves-barb-set*

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

abbreviation *rel-weakly-preserves-barbs*

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

rel-weakly-preserves-barbs $Rel\ CWB \equiv rel-preserves-binary-pred\ Rel\ (\lambda P\ a.\ P \Downarrow \langle CWB \rangle a)$

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

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

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

abbreviation *rel-weakly-reflects-barbs*

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

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

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

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

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

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

abbreviation *rel-weakly-respects-barbs*

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

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

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

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

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

fixes $Rel :: ('proc \times 'proc)\ set$
and $CWB :: ('proc, 'barbs)\ calculusWithBarbs$
assumes *preservation*: *rel-weakly-preserves-barbs* $Rel\ CWB$
shows *rel-weakly-preserves-barbs* $(Rel^=)\ CWB$
and *rel-weakly-preserves-barbs* $(Rel^+)\ CWB$
and *rel-weakly-preserves-barbs* $(Rel^*)\ CWB$
<proof>

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

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

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

```

and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes reflection: rel-weakly-reflects-barbs Rel CWB
shows rel-weakly-reflects-barbs (Rel=) CWB
and rel-weakly-reflects-barbs (Rel+) CWB
and rel-weakly-reflects-barbs (Rel*) CWB
  ⟨proof⟩

```

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

```

lemma weak-respection-of-barbs-and-closures:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes respection: rel-weakly-respects-barbs Rel CWB
shows rel-weakly-respects-barbs (Rel=) CWB
and rel-weakly-respects-barbs (symcl Rel) CWB
and rel-weakly-respects-barbs (Rel+) CWB
and rel-weakly-respects-barbs (symcl (Rel=)) CWB
and rel-weakly-respects-barbs (Rel*) CWB
and rel-weakly-respects-barbs ((symcl (Rel=))+) CWB
  ⟨proof⟩

```

```

end
theory SimulationRelations
imports ProcessCalculi
begin

```

3 Simulation Relations

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

3.1 Simulation

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

```

abbreviation weak-reduction-simulation
  :: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
where
  weak-reduction-simulation Rel Cal ≡
  ∀  $P Q P'. (P, Q) \in Rel \wedge P \mapsto Cal* P' \longrightarrow (\exists Q'. Q \mapsto Cal* Q' \wedge (P', Q') \in Rel)$ 

```

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

```

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

```

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

```

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

```

lemma *weak-barbed-simulation-and-closures*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *simulation: weak-barbed-simulation* $Rel \ CWB$
shows *weak-barbed-simulation* $(Rel^=) \ CWB$
and *weak-barbed-simulation* $(Rel^+) \ CWB$
and *weak-barbed-simulation* $(Rel^*) \ CWB$
 $\langle \text{proof} \rangle$

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

abbreviation *weak-barbed-preservation-cond*
 $:: ('proc \times 'proc) \text{ set} \Rightarrow ('proc, 'barbs) \text{ calculusWithBarbs} \Rightarrow \text{bool}$
where
weak-barbed-preservation-cond $Rel \ CWB \equiv \forall P \ Q \ a. (P, Q) \in Rel \wedge P \Downarrow \langle CWB \rangle a \longrightarrow Q \Downarrow \langle CWB \rangle a$

lemma *weak-preservation-of-barbs*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *preservation: rel-weakly-preserves-barbs* $Rel \ CWB$
shows *weak-barbed-preservation-cond* $Rel \ CWB$
 $\langle \text{proof} \rangle$

lemma *simulation-impl-equality-of-preservation-of-barbs-conditions*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *simulation: weak-reduction-simulation* $Rel \ (\text{Calculus } CWB)$
shows *rel-weakly-preserves-barbs* $Rel \ CWB = \text{weak-barbed-preservation-cond } Rel \ CWB$
 $\langle \text{proof} \rangle$

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

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

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

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

A strong strong simulation is also a weak simulation.

lemma *strong-impl-weak-reduction-simulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes *simulation: strong-reduction-simulation* $Rel \ Cal$
shows *weak-reduction-simulation* $Rel \ Cal$
 $\langle \text{proof} \rangle$

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

lemma *strong-impl-weak-barbed-simulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *simulation: strong-barbed-simulation* $Rel \ CWB$
shows *weak-barbed-simulation* $Rel \ CWB$
 $\langle \text{proof} \rangle$

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

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

lemma *strong-barbed-simulation-and-closures*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *simulation: strong-barbed-simulation* $Rel \ CWB$
shows *strong-barbed-simulation* $(Rel^=) \ CWB$
and *strong-barbed-simulation* $(Rel^+) \ CWB$
and *strong-barbed-simulation* $(Rel^*) \ CWB$
 $\langle \text{proof} \rangle$

3.2 Contrsimulation

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

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

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

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

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

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

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

and *weak-barbed-contrasimulation* (Rel^+) *CWB*
and *weak-barbed-contrasimulation* (Rel^*) *CWB*
 ⟨*proof*⟩

3.3 Coupled Simulation

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

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

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

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

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

lemma *weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation:*
fixes $Rel :: ('proc \times 'proc) set$
and $Cal :: 'proc\ processCalculus$
shows *weak-reduction-coupled-simulation* $Rel\ Cal$
 $= (weak\ reduction\ simulation\ Rel\ Cal \wedge weak\ reduction\ contrasimulation\ Rel\ Cal)$
 ⟨*proof*⟩

lemma *weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation:*
fixes $Rel :: ('proc \times 'proc) set$
and $CWB :: ('proc, 'barbs) calculusWithBarbs$
shows *weak-barbed-coupled-simulation* $Rel\ CWB$
 $= (weak\ barbed\ simulation\ Rel\ CWB \wedge weak\ barbed\ contrasimulation\ Rel\ CWB)$
 ⟨*proof*⟩

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

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

lemma *weak-barbed-coupled-simulation-and-closures:*
fixes $Rel :: ('proc \times 'proc) set$
and $CWB :: ('proc, 'barbs) calculusWithBarbs$
assumes *coupledSimulation: weak-barbed-coupled-simulation* $Rel\ CWB$
shows *weak-barbed-coupled-simulation* $(Rel^=) CWB$
and *weak-barbed-coupled-simulation* $(Rel^+) CWB$
and *weak-barbed-coupled-simulation* $(Rel^*) CWB$
 ⟨*proof*⟩

3.4 Correspondence Simulation

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

abbreviation *weak-reduction-correspondence-simulation*

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

where

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

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

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

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

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

abbreviation *weak-barbed-correspondence-simulation*

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

where

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

weak-reduction-correspondence-simulation $Rel \text{ (Calculus CWB)}$

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

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

inductive-set *cSim-cs* $:: ('proc \times 'proc) set \Rightarrow 'proc \text{ processCalculus} \Rightarrow ('proc \times 'proc) set$

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

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

where

left: $\llbracket Q \mapsto_{Cal*} Q'; (P', Q') \in Rel \rrbracket \Longrightarrow (P', Q) \in \text{cSim-cs } Rel \text{ Cal} \mid$

right: $\llbracket P \mapsto_{Cal*} P'; (Q, P) \in Rel \rrbracket \Longrightarrow (P', Q) \in \text{cSim-cs } Rel \text{ Cal} \mid$

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

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

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

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

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

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

and $\forall P \ Q. (P, Q) \in Rel \longrightarrow (P, Q) \in \text{cSim-cs } Rel \text{ Cal} \wedge (Q, P) \in \text{cSim-cs } Rel \text{ Cal}$

<proof>

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

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

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

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

shows *weak-barbed-coupled-simulation* $(\text{cSim-cs } Rel \text{ (Calculus CWB)}) \text{ CWB}$

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

<proof>

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

fixes $Cal :: 'proc \text{ processCalculus}$

and $P \ Q \ R :: 'proc$

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

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

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

and $A3: \text{weak-reduction-simulation } Rel \text{ Cal}$

and $A4: \text{trans } Rel$

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

<proof>

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

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

lemma *weak-barbed-correspondence-simulation-and-closures:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *corrSim: weak-barbed-correspondence-simulation Rel CWB*
shows *weak-barbed-correspondence-simulation (Rel⁻) CWB*
and *weak-barbed-correspondence-simulation (Rel⁺) CWB*
and *weak-barbed-correspondence-simulation (Rel^{*}) CWB*
<proof>

3.5 Bisimulation

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

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

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

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

A symmetric weak simulation is a weak bisimulation.

lemma *symm-weak-reduction-simulation-is-bisimulation:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes *sym Rel*
and *weak-reduction-simulation Rel Cal*
shows *weak-reduction-bisimulation Rel Cal*
<proof>

lemma *symm-weak-barbed-simulation-is-bisimulation:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes *sym Rel*
and *weak-barbed-simulation Rel Cal*
shows *weak-barbed-bisimulation Rel Cal*
<proof>

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

lemma *weak-reduction-simulations-impl-bisimulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes $sim: \text{ weak-reduction-simulation } Rel \text{ } Cal$
and $simInv: \text{ weak-reduction-simulation } (Rel^{-1}) \text{ } Cal$
shows $\text{ weak-reduction-bisimulation } Rel \text{ } Cal$
 $\langle proof \rangle$

lemma *weak-reduction-bisimulations-impl-inverse-is-simulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes $bisim: \text{ weak-reduction-bisimulation } Rel \text{ } Cal$
shows $\text{ weak-reduction-simulation } (Rel^{-1}) \text{ } Cal$
 $\langle proof \rangle$

lemma *weak-reduction-simulations-iff-bisimulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
shows $(\text{ weak-reduction-simulation } Rel \text{ } Cal \wedge \text{ weak-reduction-simulation } (Rel^{-1}) \text{ } Cal)$
 $= \text{ weak-reduction-bisimulation } Rel \text{ } Cal$
 $\langle proof \rangle$

lemma *weak-barbed-simulations-iff-bisimulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
shows $(\text{ weak-barbed-simulation } Rel \text{ } CWB \wedge \text{ weak-barbed-simulation } (Rel^{-1}) \text{ } CWB)$
 $= \text{ weak-barbed-bisimulation } Rel \text{ } CWB$
 $\langle proof \rangle$

A weak bisimulation is a weak correspondence simulation.

lemma *weak-reduction-bisimulation-is-correspondence-simulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes $bisim: \text{ weak-reduction-bisimulation } Rel \text{ } Cal$
shows $\text{ weak-reduction-correspondence-simulation } Rel \text{ } Cal$
 $\langle proof \rangle$

lemma *weak-barbed-bisimulation-is-correspondence-simulation*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes $bisim: \text{ weak-barbed-bisimulation } Rel \text{ } CWB$
shows $\text{ weak-barbed-correspondence-simulation } Rel \text{ } CWB$
 $\langle proof \rangle$

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

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

lemma *weak-barbed-bisimulation-and-closures*:
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$

assumes *bisim*: weak-barbed-bisimulation *Rel* *CWB*
shows weak-barbed-bisimulation (*Rel*⁼) *CWB*
and weak-barbed-bisimulation (*symcl* *Rel*) *CWB*
and weak-barbed-bisimulation (*Rel*⁺) *CWB*
and weak-barbed-bisimulation (*symcl* (*Rel*⁼)) *CWB*
and weak-barbed-bisimulation (*Rel*^{*}) *CWB*
and weak-barbed-bisimulation ((*symcl* (*Rel*⁼))⁺) *CWB*
 ⟨*proof*⟩

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

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

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

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

A symmetric strong simulation is a strong bisimulation.

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

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

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

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

lemma *strong-reduction-bisimulations-impl-inverse-is-simulation*:
fixes *Rel* :: ('*proc* × '*proc*) set
and *Cal* :: '*proc* processCalculus
assumes *bisim*: strong-reduction-bisimulation *Rel* *Cal*
shows strong-reduction-simulation (*Rel*⁻¹) *Cal*
 ⟨*proof*⟩

lemma *strong-reduction-simulations-iff-bisimulation:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
shows $(\text{strong-reduction-simulation } Rel \text{ } Cal \wedge \text{strong-reduction-simulation } (Rel^{-1}) \text{ } Cal)$
 $= \text{strong-reduction-bisimulation } Rel \text{ } Cal$
 $\langle \text{proof} \rangle$

lemma *strong-barbed-simulations-iff-bisimulation:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
shows $(\text{strong-barbed-simulation } Rel \text{ } CWB \wedge \text{strong-barbed-simulation } (Rel^{-1}) \text{ } CWB)$
 $= \text{strong-barbed-bisimulation } Rel \text{ } CWB$
 $\langle \text{proof} \rangle$

A strong bisimulation is a weak bisimulation.

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

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

lemma *strong-impl-weak-barbed-bisimulation:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes $\text{bisim}: \text{strong-barbed-bisimulation } Rel \text{ } CWB$
shows $\text{weak-barbed-bisimulation } Rel \text{ } CWB$
 $\langle \text{proof} \rangle$

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

lemma *strong-reduction-bisimulation-and-closures:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $Cal :: 'proc \text{ processCalculus}$
assumes $\text{bisim}: \text{strong-reduction-bisimulation } Rel \text{ } Cal$
shows $\text{strong-reduction-bisimulation } (Rel^=) \text{ } Cal$
and $\text{strong-reduction-bisimulation } (\text{symcl } Rel) \text{ } Cal$
and $\text{strong-reduction-bisimulation } (Rel^+) \text{ } Cal$
and $\text{strong-reduction-bisimulation } (\text{symcl } (Rel^=)) \text{ } Cal$
and $\text{strong-reduction-bisimulation } (Rel^*) \text{ } Cal$
and $\text{strong-reduction-bisimulation } ((\text{symcl } (Rel^=))^+) \text{ } Cal$
 $\langle \text{proof} \rangle$

lemma *strong-barbed-bisimulation-and-closures:*
fixes $Rel :: ('proc \times 'proc) \text{ set}$
and $CWB :: ('proc, 'barbs) \text{ calculusWithBarbs}$
assumes $\text{bisim}: \text{strong-barbed-bisimulation } Rel \text{ } CWB$
shows $\text{strong-barbed-bisimulation } (Rel^=) \text{ } CWB$
and $\text{strong-barbed-bisimulation } (\text{symcl } Rel) \text{ } CWB$
and $\text{strong-barbed-bisimulation } (Rel^+) \text{ } CWB$
and $\text{strong-barbed-bisimulation } (\text{symcl } (Rel^=)) \text{ } CWB$
and $\text{strong-barbed-bisimulation } (Rel^*) \text{ } CWB$
and $\text{strong-barbed-bisimulation } ((\text{symcl } (Rel^=))^+) \text{ } CWB$
 $\langle \text{proof} \rangle$

3.6 Step Closure of Relations

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

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

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

Applying the steps closure twice does not change the relation.

lemma *steps-closure-of-steps-closure*:
fixes *Rel* :: ('a × 'a) set
and *Cal* :: 'a processCalculus
shows *stepsClosure (stepsClosure Rel Cal) Cal = stepsClosure Rel Cal*
⟨proof⟩

The steps closure is a preorder.

lemma *stepsClosure-refl*:
fixes *Rel* :: ('a × 'a) set
and *Cal* :: 'a processCalculus
shows *refl (stepsClosure Rel Cal)*
⟨proof⟩

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

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

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

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

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

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

```

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-contrasimulation (stepsClosure Rel Cal) Cal
⟨proof⟩

```

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

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

```

fixes Rel :: ('a × 'a) set
and Cal :: 'a processCalculus
assumes sim: weak-reduction-simulation Rel Cal
and con: weak-reduction-contrasimulation (Rel-1) Cal
shows weak-reduction-coupled-simulation (stepsClosure Rel Cal) Cal
⟨proof⟩

```

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

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

```

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

```

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

```

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

```

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

```

fixes Rel :: ('a × 'a) set
and Cal :: 'a processCalculus
and P Q :: 'a
assumes A1: P  $\mathcal{R} \mapsto \langle \text{Rel}^{-1}, \text{Cal} \rangle$  Q
and A2: weak-reduction-contrasimulation (Rel-1) Cal
shows  $\exists Q'. Q \mapsto \text{Cal}^* Q' \wedge (P, Q') \in \text{Rel}^*$ 
⟨proof⟩

```

end

theory Encodings

imports ProcessCalculi

begin

4 Encodings

In the simplest case an encoding from a source into a target language is a mapping from source into target terms. Encodability criteria describe properties on such mappings. To analyse encodability criteria we map them on conditions on relations between source and target terms. More precisely, we consider relations on pairs of the disjoint union of source and target terms. We denote this disjoint union of source and target terms by Proc.

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

definition STCal

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

where

STCal Source Target \equiv
 (\Downarrow Reductions = $\lambda P P'$.
 ($\exists SP SP'. P = \text{SourceTerm } SP \wedge P' = \text{SourceTerm } SP' \wedge \text{Reductions Source } SP SP'$) \vee
 ($\exists TP TP'. P = \text{TargetTerm } TP \wedge P' = \text{TargetTerm } TP' \wedge \text{Reductions Target } TP TP'$))

definition STCalWB

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

where

STCalWB Source Target \equiv
 (\Downarrow Calculus = STCal (calculusWithBarbs.Calculus Source) (calculusWithBarbs.Calculus Target),
 HasBarb = $\lambda P a. (\exists SP. P = \text{SourceTerm } SP \wedge (\text{calculusWithBarbs.HasBarb Source } SP a)) \vee$
 ($\exists TP. P = \text{TargetTerm } TP \wedge (\text{calculusWithBarbs.HasBarb Target } TP a)$))

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

locale encoding =

fixes Source :: 'procS processCalculus
and Target :: 'procT processCalculus
and Enc :: 'procS \Rightarrow 'procT

begin

abbreviation enc :: 'procS \Rightarrow 'procT (\llbracket - \rrbracket [65] 70) **where**
 $\llbracket S \rrbracket \equiv \text{Enc } S$

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

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

abbreviation getSource

:: 'procS \Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \in S - [70, 70] 80)
where
 $S \in S P \equiv (P = \text{SourceTerm } S)$

abbreviation getTarget

:: 'procT \Rightarrow ('procS, 'procT) Proc \Rightarrow bool (- \in T - [70, 70] 80)
where
 $T \in T P \equiv (P = \text{TargetTerm } T)$

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

abbreviation stepST

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

lemma stepST-STCal-step:

fixes P P' :: ('procS, 'procT) Proc
shows $P \mapsto$ (STCal Source Target) $P' = P \mapsto$ ST P'
 (proof)

lemma *STStep-step*:

fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) Proc$
shows $SourceTerm\ S \mapsto_{ST} P' = (\exists S'. S' \in S\ P' \wedge S \mapsto_{Source} S')$
and $TargetTerm\ T \mapsto_{ST} P' = (\exists T'. T' \in T\ P' \wedge T \mapsto_{Target} T')$
 $\langle proof \rangle$

lemma *STCal-step*:

fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) Proc$
shows $SourceTerm\ S \mapsto_{(STCal\ Source\ Target)} P' = (\exists S'. S' \in S\ P' \wedge S \mapsto_{Source} S')$
and $TargetTerm\ T \mapsto_{(STCal\ Source\ Target)} P' = (\exists T'. T' \in T\ P' \wedge T \mapsto_{Target} T')$
 $\langle proof \rangle$

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

abbreviation *stepsST*

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

lemma *STSteps-steps*:

fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) Proc$
shows $SourceTerm\ S \mapsto_{ST*} P' = (\exists S'. S' \in S\ P' \wedge S \mapsto_{Source*} S')$
and $TargetTerm\ T \mapsto_{ST*} P' = (\exists T'. T' \in T\ P' \wedge T \mapsto_{Target*} T')$
 $\langle proof \rangle$

lemma *STCal-steps*:

fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) Proc$
shows $SourceTerm\ S \mapsto_{(STCal\ Source\ Target)*} P' = (\exists S'. S' \in S\ P' \wedge S \mapsto_{Source*} S')$
and $TargetTerm\ T \mapsto_{(STCal\ Source\ Target)*} P' = (\exists T'. T' \in T\ P' \wedge T \mapsto_{Target*} T')$
 $\langle proof \rangle$

lemma *stepsST-STCal-steps*:

fixes $P\ P' :: ('procS, 'procT) Proc$
shows $P \mapsto_{(STCal\ Source\ Target)*} P' = P \mapsto_{ST*} P'$
 $\langle proof \rangle$

lemma *stepsST-refl*:

fixes $P :: ('procS, 'procT) Proc$
shows $P \mapsto_{ST*} P$
 $\langle proof \rangle$

lemma *stepsST-add*:

fixes $P\ Q\ R :: ('procS, 'procT) Proc$
assumes $A1: P \mapsto_{ST*} Q$
and $A2: Q \mapsto_{ST*} R$
shows $P \mapsto_{ST*} R$
 $\langle proof \rangle$

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

abbreviation *divergentST*

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

lemma *STCal-divergent*:

fixes $S :: 'procS$
and $T :: 'procT$
shows $SourceTerm S \mapsto (STCal Source Target)\omega = S \mapsto (Source)\omega$
and $TargetTerm T \mapsto (STCal Source Target)\omega = T \mapsto (Target)\omega$
 $\langle proof \rangle$

lemma *divergentST-STCal-divergent*:

fixes $P :: ('procS, 'procT) Proc$
shows $P \mapsto (STCal Source Target)\omega = P \mapsto ST\omega$
 $\langle proof \rangle$

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(enc S) for all source terms S.

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

abbreviation *enc-preserves-binary-pred*

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

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

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

abbreviation *enc-reflects-binary-pred*

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

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

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

abbreviation *enc-respects-binary-pred*

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

end

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

locale *encoding-wrt-barbs* =

$encoding Source Target Enc$
for $Source :: 'procS processCalculus$
and $Target :: 'procT processCalculus$
and $Enc :: 'procS \Rightarrow 'procT +$
fixes $SWB :: ('procS, 'barbs) calculusWithBarbs$
and $TWB :: ('procT, 'barbs) calculusWithBarbs$
assumes $calS: calculusWithBarbs.Calculus SWB = Source$
and $calT: calculusWithBarbs.Calculus TWB = Target$
begin

lemma *STCalWB-STCal*:

shows *Calculus* (*STCalWB SWB TWB*) = *STCal Source Target*
 ⟨*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*

$:: ('procS, 'procT) Proc \Rightarrow 'barbs \Rightarrow bool (-\downarrow.- [70, 70] 80)$

where

$P\downarrow.a \equiv (\exists S. S \in S P \wedge S\downarrow\langle SWB \rangle a) \vee (\exists T. T \in T P \wedge T\downarrow\langle TWB \rangle a)$

lemma *STCalWB-hasBarbST*:

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

and $a :: 'barbs$

shows $P\downarrow\langle STCalWB SWB TWB \rangle a = P\downarrow.a$

⟨*proof*⟩

lemma *preservation-of-barbs-in-barbed-encoding*:

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

and $P Q :: ('procS, 'procT) Proc$

and $a :: 'barbs$

assumes *preservation: rel-preserves-barbs Rel* (*STCalWB SWB TWB*)

and $rel: (P, Q) \in Rel$

and $barb: P\downarrow.a$

shows $Q\downarrow.a$

⟨*proof*⟩

lemma *reflection-of-barbs-in-barbed-encoding*:

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

and $P Q :: ('procS, 'procT) Proc$

and $a :: 'barbs$

assumes *reflection: rel-reflects-barbs Rel* (*STCalWB SWB TWB*)

and $rel: (P, Q) \in Rel$

and $barb: Q\downarrow.a$

shows $P\downarrow.a$

⟨*proof*⟩

lemma *respection-of-barbs-in-barbed-encoding*:

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

and $P Q :: ('procS, 'procT) Proc$

and $a :: 'barbs$

assumes *respection: rel-respects-barbs Rel* (*STCalWB SWB TWB*)

and $rel: (P, Q) \in Rel$

shows $P\downarrow.a = Q\downarrow.a$

⟨*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*

$:: ('procS, 'procT) Proc \Rightarrow 'barbs \Rightarrow bool (-\downarrow.- [70, 70] 80)$

where

$P\downarrow.a \equiv (\exists S. S \in S P \wedge S\downarrow\langle SWB \rangle a) \vee (\exists T. T \in T P \wedge T\downarrow\langle TWB \rangle a)$

lemma *STCalWB-reachesBarbST*:

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

and $a :: 'barbs$

shows $P\downarrow\langle STCalWB SWB TWB \rangle a = P\downarrow.a$

⟨*proof*⟩

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

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

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

end

end

theory *SourceTargetRelation*

imports *Encodings SimulationRelations*

begin

5 Relation between Source and Target Terms

5.1 Relations Induced by the Encoding Function

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

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

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

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

target: $(\text{TargetTerm } T, \text{TargetTerm } T) \in \text{indRelRPO} \mid$
trans: $\llbracket (P, Q) \in \text{indRelRPO}; (Q, R) \in \text{indRelRPO} \rrbracket \implies (P, R) \in \text{indRelRPO}$

abbreviation (in encoding) *indRelRPOinfix* ::

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

where

$P \lesssim \llbracket \cdot \rrbracket R Q \equiv (P, Q) \in \text{indRelRPO}$

lemma (in encoding) *indRelRPO-refl*:

shows *refl indRelRPO*

$\langle \text{proof} \rangle$

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

shows *preorder indRelRPO*

$\langle \text{proof} \rangle$

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

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

$\langle \text{proof} \rangle$

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

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

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

and *PredB* :: $((\text{'procS}, \text{'procT}) \text{Proc} \times (\text{'procS}, \text{'procT}) \text{Proc}) \Rightarrow \text{bool}$

shows $\text{PredA } \text{indRelR} \implies \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge \text{PredA } \text{Rel}$

and $\forall (P, Q) \in \text{indRelR}. \text{PredB } (P, Q)$

$\implies \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{PredB } (P, Q))$

$\langle \text{proof} \rangle$

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

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

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

assumes *encRel*: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

and *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) *indRelR-iff-exists-source-target-relation*:

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

shows $(\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q))$

$= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)))$

$\langle \text{proof} \rangle$

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

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

assumes *reflCond*: $\forall P. \text{Pred } (P, P)$

and *transCond*: $\forall P Q R. \text{Pred } (P, Q) \wedge \text{Pred } (Q, R) \longrightarrow \text{Pred } (P, R)$

shows $(\forall (P, Q) \in \text{indRelR}. \text{Pred } (P, Q)) = (\forall (P, Q) \in \text{indRelRPO}. \text{Pred } (P, Q))$

$\langle \text{proof} \rangle$

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

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

shows $(\forall (P, Q) \in \text{indRelRPO}. \text{Pred } (P, Q)) = (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge (\forall (P, Q) \in \text{Rel}. \text{Pred } (P, Q)) \wedge \text{preorder } \text{Rel})$

$\langle \text{proof} \rangle$

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

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

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

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

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

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

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

lemma (in *encoding*) *enc-preserves-pred-iff-indRelRPO-preserves-pred*:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-preserves-pred Pred = rel-preserves-pred indRelRPO Pred$
 $\langle proof \rangle$

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

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

lemma (in *encoding*) *enc-reflects-pred-iff-indRelRPO-reflects-pred*:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-reflects-pred Pred = rel-reflects-pred indRelRPO Pred$
 $\langle proof \rangle$

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

lemma (in *encoding*) *enc-respects-binary-pred-iff-indRelR-respects-binary-pred*:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$
shows $enc-respects-binary-pred Pred = rel-respects-binary-pred indRelR Pred$
 $\langle proof \rangle$

lemma (in *encoding*) *enc-respects-pred-iff-indRelRPO-respects-pred*:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-respects-pred\ Pred = rel-respects-pred\ indRelRPO\ Pred$
 <proof>

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

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

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

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

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

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

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

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

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

inductive-set (in *encoding*) *indRelL*
 :: (((*'procS*, *'procT*) *Proc*) × ((*'procS*, *'procT*) *Proc*)) *set*
where
encL: (*TargetTerm* ($\llbracket S \rrbracket$), *SourceTerm* *S*) ∈ *indRelL*

abbreviation (in *encoding*) *indRelLinf* ::
 (*'procS*, *'procT*) *Proc* ⇒ (*'procS*, *'procT*) *Proc* ⇒ *bool* (- $\mathcal{R}[\cdot]L$ - [75, 75] 80)
where
P $\mathcal{R}[\cdot]L$ *Q* ≡ (*P*, *Q*) ∈ *indRelL*

inductive-set (in *encoding*) *indRelLPO*
 :: (((*'procS*, *'procT*) *Proc*) × ((*'procS*, *'procT*) *Proc*)) *set*
where
encL: (*TargetTerm* ($\llbracket S \rrbracket$), *SourceTerm* *S*) ∈ *indRelLPO* |
source: (*SourceTerm* *S*, *SourceTerm* *S*) ∈ *indRelLPO* |
target: (*TargetTerm* *T*, *TargetTerm* *T*) ∈ *indRelLPO* |
trans: $\llbracket (P, Q) \in \text{indRelLPO}; (Q, R) \in \text{indRelLPO} \rrbracket \implies (P, R) \in \text{indRelLPO}$

abbreviation (in *encoding*) *indRelLPOinf* ::
 (*'procS*, *'procT*) *Proc* ⇒ (*'procS*, *'procT*) *Proc* ⇒ *bool* (- $\lesssim[\cdot]L$ - [75, 75] 80)
where
P $\lesssim[\cdot]L$ *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 $rel\text{-respects-pred } indRelR \text{ Pred} = rel\text{-respects-pred } indRelL \text{ Pred}$
 ⟨proof⟩

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

lemma (in *encoding*) $indRelR\text{-cond-preservation-iff-}indRelL\text{-cond-reflection}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ Pred})$
 ⟨proof⟩

lemma (in *encoding*) $indRelR\text{-cond-binary-preservation-iff-}indRelL\text{-cond-binary-reflection}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-preserves-binary-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$
 $\wedge rel\text{-reflects-binary-pred } Rel \text{ Pred})$
 ⟨proof⟩

lemma (in *encoding*) $indRelR\text{-cond-reflection-iff-}indRelL\text{-cond-preservation}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-preserves-pred } Rel \text{ Pred})$
 ⟨proof⟩

lemma (in *encoding*) $indRelR\text{-cond-binary-reflection-iff-}indRelL\text{-cond-binary-preservation}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-reflects-binary-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$
 $\wedge rel\text{-preserves-binary-pred } Rel \text{ Pred})$
 ⟨proof⟩

lemma (in *encoding*) $indRelR\text{-cond-respection-iff-}indRelL\text{-cond-respection}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel) \wedge rel\text{-respects-pred } Rel \text{ Pred})$
 ⟨proof⟩

lemma (in *encoding*) $indRelR\text{-cond-binary-respection-iff-}indRelL\text{-cond-binary-respection}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool$
shows $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge rel\text{-respects-binary-pred } Rel \text{ Pred})$
 $= (\exists Rel. (\forall S. (TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in Rel)$
 $\wedge rel\text{-respects-binary-pred } Rel \text{ Pred})$
 ⟨proof⟩

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

lemma (in *encoding*) $enc\text{-preserves-pred-iff-}indRelL\text{-reflects-pred}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc\text{-preserves-pred } Pred = rel\text{-reflects-pred } indRelL \text{ Pred}$
 ⟨proof⟩

lemma (in *encoding*) $enc\text{-reflects-pred-iff-}indRelL\text{-preserves-pred}$:
fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc\text{-reflects-pred } Pred = rel\text{-preserves-pred } indRelL \text{ Pred}$
 ⟨proof⟩

lemma (in *encoding*) $enc\text{-respects-pred-iff-}indRelL\text{-respects-pred}$:

fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-respects-pred\ Pred = rel-respects-pred\ indRelL\ Pred$
 $\langle proof \rangle$

An encoding preserves, reflects, or respects a predicate iff there exists a relation, namely $indRelL$, that relates literal translations with their source terms and reflects, preserves, or respects this predicate.

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

fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-preserves-pred\ Pred$
 $= (\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-reflects-pred\ Rel\ Pred)$
 $\langle proof \rangle$

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

fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-reflects-pred\ Pred$
 $= (\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-preserves-pred\ Rel\ Pred)$
 $\langle proof \rangle$

lemma (**in** *encoding*) *enc-respects-pred-iff-source-target-rel-respects-pred-encL*:

fixes $Pred :: ('procS, 'procT) Proc \Rightarrow bool$
shows $enc-respects-pred\ Pred$
 $= (\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-respects-pred\ Rel\ Pred)$
 $\langle proof \rangle$

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

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

$:: ((('procS, 'procT) Proc) \times ((('procS, 'procT) Proc))\ set$
where
 $encR: (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRel \mid$
 $encL: (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in indRel$

abbreviation (**in** *encoding*) *indRelInfix* ::

$('procS, 'procT) Proc \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$ ($- \mathcal{R}[\cdot]$ - [75, 75] 80)
where
 $P \mathcal{R}[\cdot] Q \equiv (P, Q) \in indRel$

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

shows $sym\ indRel$
 $\langle proof \rangle$

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

$:: ((('procS, 'procT) Proc) \times ((('procS, 'procT) Proc))\ set$
where
 $encR: (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelEQ \mid$
 $encL: (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in indRelEQ \mid$
 $target: (TargetTerm\ T, TargetTerm\ T) \in indRelEQ \mid$
 $trans: \llbracket (P, Q) \in indRelEQ; (Q, R) \in indRelEQ \rrbracket \Longrightarrow (P, R) \in indRelEQ$

abbreviation (**in** *encoding*) *indRelEQinfix* ::

$('procS, 'procT) Proc \Rightarrow ('procS, 'procT) Proc \Rightarrow bool$ ($- \sim[\cdot]$ - [75, 75] 80)
where
 $P \sim[\cdot] Q \equiv (P, Q) \in indRelEQ$

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

shows $refl\ indRelEQ$
 $\langle proof \rangle$

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

shows $preorder\ indRelEQ$

<proof>

lemma (in *encoding*) *indRelEQ-symm*:
shows *sym indRelEQ*
<proof>

lemma (in *encoding*) *indRelEQ-is-equivalence*:
shows *equivalence indRelEQ*
<proof>

lemma (in *encoding*) *refl-trans-closure-of-indRel*:
shows *indRelEQ = indRel**
<proof>

lemma (in *encoding*) *refl-symm-trans-closure-of-indRel*:
shows *indRelEQ = (symcl (indRel⁼))⁺*
<proof>

lemma (in *encoding*) *symm-closure-of-indRelR*:
shows *indRel = symcl indRelR*
and *indRelEQ = (symcl (indRelR⁼))⁺*
<proof>

lemma (in *encoding*) *symm-closure-of-indRelL*:
shows *indRel = symcl indRelL*
and *indRelEQ = (symcl (indRelL⁼))⁺*
<proof>

The relation *indRel* is a combination of *indRelL* and *indRelR*. *indRel* respects a predicate iff *indRelR* (or *indRelL*) respects it.

lemma (in *encoding*) *indRel-respects-pred-iff-indRelR-respects-pred*:
fixes *Pred :: ('procS, 'procT) Proc ⇒ bool*
shows *rel-respects-pred indRel Pred = rel-respects-pred indRelR Pred*
<proof>

lemma (in *encoding*) *indRel-respects-binary-pred-iff-indRelR-respects-binary-pred*:
fixes *Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool*
shows *rel-respects-binary-pred indRel Pred = rel-respects-binary-pred indRelR Pred*
<proof>

lemma (in *encoding*) *indRel-cond-respection-iff-indRelR-cond-respection*:
fixes *Pred :: ('procS, 'procT) Proc ⇒ bool*
shows $(\exists Rel.$
 $(\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel \wedge (TargetTerm\ ([S]), SourceTerm\ S) \in Rel)$
 $\wedge rel-respects-pred\ Rel\ Pred)$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel) \wedge rel-respects-pred\ Rel\ Pred)$
<proof>

lemma (in *encoding*) *indRel-cond-binary-respection-iff-indRelR-cond-binary-respection*:
fixes *Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool*
shows $(\exists Rel.$
 $(\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel \wedge (TargetTerm\ ([S]), SourceTerm\ S) \in Rel)$
 $\wedge rel-respects-binary-pred\ Rel\ Pred)$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel)$
 $\wedge rel-respects-binary-pred\ Rel\ Pred)$
<proof>

An encoding respects a predicate iff *indRel* respects this predicate.

lemma (in *encoding*) *enc-respects-pred-iff-indRel-respects-pred*:
fixes *Pred :: ('procS, 'procT) Proc ⇒ bool*

shows *enc-respects-pred* $Pred = rel-respects-pred\ indRel\ Pred$
 ⟨proof⟩

An encoding respects a predicate iff there exists a relation, namely *indRel*, that relates source terms and their literal translations in both directions and respects this predicate.

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

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

shows *enc-respects-pred* $Pred$

= $(\exists Rel.$

$(\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$
 $\wedge rel-respects-pred\ Rel\ Pred)$

⟨proof⟩

5.2 Relations Induced by the Encoding and a Relation on Target Terms

Some encodability like e.g. operational correspondence are defined w.r.t. a relation on target terms. To analyse such criteria we include the respective target term relation in the considered relation on the disjoint union of source and target terms.

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

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

for $TRel :: ('procT \times 'procT)\ set$

where

encR: $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelRT\ TRel \mid$

target: $(T1, T2) \in TRel \Longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in indRelRT\ TRel$

abbreviation (in *encoding*) *indRelRTinfix*

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

$(- \mathcal{R}[\cdot]RT <-> - [75, 75, 75] 80)$

where

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

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

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

for $TRel :: ('procT \times 'procT)\ set$

where

encR: $(SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in indRelRTPO\ TRel \mid$

source: $(SourceTerm\ S, SourceTerm\ S) \in indRelRTPO\ TRel \mid$

target: $(T1, T2) \in TRel \Longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in indRelRTPO\ TRel \mid$

trans: $\llbracket (P, Q) \in indRelRTPO\ TRel; (Q, R) \in indRelRTPO\ TRel \rrbracket \Longrightarrow (P, R) \in indRelRTPO\ TRel$

abbreviation (in *encoding*) *indRelRTPOinfix*

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

$(- \lesssim[\cdot]RT <-> - [75, 75, 75] 80)$

where

$P \lesssim[\cdot]RT < TRel > Q \equiv (P, Q) \in indRelRTPO\ TRel$

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

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

assumes *refl*: *refl* $TRel$

shows *refl* $(indRelRTPO\ TRel)$

⟨proof⟩

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

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

assumes *refl*: *refl* $TRel$

shows $indRelRTPO\ TRel = (indRelRT\ TRel)^*$

⟨proof⟩

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

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

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $TP\ TQ :: 'procT$
shows $(TP, TQ) \in TRel^+ \implies TargetTerm\ TP \lesssim_{[\cdot]RT<TRel>} TargetTerm\ TQ$
 $\langle proof \rangle$

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

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

fixes $Pred :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \implies bool$
shows $(\forall (P, Q) \in indRelR. Pred\ (P, Q)) = (\forall TRel. (\forall (TP, TQ) \in TRel. Pred\ (TargetTerm\ TP, TargetTerm\ TQ))) \iff (\forall (P, Q) \in indRelRT\ TRel. Pred\ (P, Q))$
 $\langle proof \rangle$

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

fixes $Pred :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \implies bool$
shows $(\forall TRel. (\forall (TP, TQ) \in TRel. Pred\ (TargetTerm\ TP, TargetTerm\ TQ))) \iff (\forall (P, Q) \in indRelRT\ TRel. Pred\ (P, Q))$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ ([S])) \in Rel) \wedge (\forall (P, Q) \in Rel. Pred\ (P, Q)))$
 $\langle proof \rangle$

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

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

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

fixes $Pred :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \implies bool$
assumes $\forall P. Pred\ (P, P)$
and $\forall P\ Q\ R. Pred\ (P, Q) \wedge Pred\ (Q, R) \implies Pred\ (P, R)$
shows $(\forall (P, Q) \in indRelR. Pred\ (P, Q)) = (\forall TRel. (\forall (TP, TQ) \in TRel. Pred\ (TargetTerm\ TP, TargetTerm\ TQ))) \iff (\forall (P, Q) \in indRelRTPO\ TRel. Pred\ (P, Q))$
 $\langle proof \rangle$

The relation $indRelLT$ includes $TRel$ and relates literal translations and their source terms.

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

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

abbreviation (in *encoding*) *indRelLTinfix*

$:: ('procS, 'procT) Proc \implies ('procT \times 'procT) \text{ set} \implies ('procS, 'procT) Proc \implies bool$
 $(- \mathcal{R}[\cdot]LT<-> - [75, 75, 75] 80)$
where
 $P \mathcal{R}[\cdot]LT<TRel> Q \equiv (P, Q) \in indRelLT\ TRel$

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

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

for $TRel :: ('procT \times 'procT) \text{ set}$
where
encL: $(TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in indRelLTPO \ TRel \ |$
source: $(SourceTerm S, SourceTerm S) \in indRelLTPO \ TRel \ |$
target: $(T1, T2) \in TRel \implies (TargetTerm T1, TargetTerm T2) \in indRelLTPO \ TRel \ |$
trans: $\llbracket (P, Q) \in indRelLTPO \ TRel; (Q, R) \in indRelLTPO \ TRel \rrbracket \implies (P, R) \in indRelLTPO \ TRel$

abbreviation (in encoding) indRelLTPOinfix
 $:: ('procS, 'procT) \text{ Proc} \Rightarrow ('procT \times 'procT) \text{ set} \Rightarrow ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
 $(- \lesssim \llbracket \cdot \rrbracket LT <-> - [75, 75, 75] 80)$
where
 $P \lesssim \llbracket \cdot \rrbracket LT <TRel> Q \equiv (P, Q) \in indRelLTPO \ TRel$

lemma (in encoding) indRelLTPO-refl:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *refl*: $refl \ TRel$
shows *refl* $(indRelLTPO \ TRel)$
 $\langle proof \rangle$

lemma (in encoding) refl-trans-closure-of-indRelLT:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *refl*: $refl \ TRel$
shows $indRelLTPO \ TRel = (indRelLT \ TRel)^*$
 $\langle proof \rangle$

inductive-set (in encoding) indRelT
 $:: ('procT \times 'procT) \text{ set} \Rightarrow (((('procS, 'procT) \text{ Proc}) \times ((('procS, 'procT) \text{ Proc}))) \text{ set}$
for $TRel :: ('procT \times 'procT) \text{ set}$
where
encR: $(SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in indRelT \ TRel \ |$
encL: $(TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in indRelT \ TRel \ |$
target: $(T1, T2) \in TRel \implies (TargetTerm T1, TargetTerm T2) \in indRelT \ TRel$

abbreviation (in encoding) indRelTinfix
 $:: ('procS, 'procT) \text{ Proc} \Rightarrow ('procT \times 'procT) \text{ set} \Rightarrow ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
 $(- \mathcal{R} \llbracket \cdot \rrbracket T <-> - [75, 75, 75] 80)$
where
 $P \mathcal{R} \llbracket \cdot \rrbracket T <TRel> Q \equiv (P, Q) \in indRelT \ TRel$

lemma (in encoding) indRelT-symm:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *symm*: $sym \ TRel$
shows *sym* $(indRelT \ TRel)$
 $\langle proof \rangle$

inductive-set (in encoding) indRelTEQ
 $:: ('procT \times 'procT) \text{ set} \Rightarrow (((('procS, 'procT) \text{ Proc}) \times ((('procS, 'procT) \text{ Proc}))) \text{ set}$
for $TRel :: ('procT \times 'procT) \text{ set}$
where
encR: $(SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in indRelTEQ \ TRel \ |$
encL: $(TargetTerm (\llbracket S \rrbracket), SourceTerm S) \in indRelTEQ \ TRel \ |$
target: $(T1, T2) \in TRel \implies (TargetTerm T1, TargetTerm T2) \in indRelTEQ \ TRel \ |$
trans: $\llbracket (P, Q) \in indRelTEQ \ TRel; (Q, R) \in indRelTEQ \ TRel \rrbracket \implies (P, R) \in indRelTEQ \ TRel$

abbreviation (in encoding) indRelTEQinfix
 $:: ('procS, 'procT) \text{ Proc} \Rightarrow ('procT \times 'procT) \text{ set} \Rightarrow ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
 $(- \sim \llbracket \cdot \rrbracket T <-> - [75, 75, 75] 80)$
where
 $P \sim \llbracket \cdot \rrbracket T <TRel> Q \equiv (P, Q) \in indRelTEQ \ TRel$

lemma (in encoding) indRelTEQ-refl:

fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $refl: refl\ TRel$
shows $refl\ (indRelTEQ\ TRel)$
 $\langle proof \rangle$

lemma (*in encoding*) $indRelTEQ\text{-}symm$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $symm: sym\ TRel$
shows $sym\ (indRelTEQ\ TRel)$
 $\langle proof \rangle$

lemma (*in encoding*) $refl\text{-}trans\text{-}closure\text{-}of\text{-}indRelT$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $refl: refl\ TRel$
shows $indRelTEQ\ TRel = (indRelT\ TRel)^*$
 $\langle proof \rangle$

lemma (*in encoding*) $refl\text{-}symm\text{-}trans\text{-}closure\text{-}of\text{-}indRelT$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $refl: refl\ TRel$
and $symm: sym\ TRel$
shows $indRelTEQ\ TRel = (symcl\ ((indRelT\ TRel)=))^+$
 $\langle proof \rangle$

lemma (*in encoding*) $symm\text{-}closure\text{-}of\text{-}indRelRT$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $refl: refl\ TRel$
and $symm: sym\ TRel$
shows $indRelT\ TRel = symcl\ (indRelRT\ TRel)$
and $indRelTEQ\ TRel = (symcl\ ((indRelRT\ TRel)=))^+$
 $\langle proof \rangle$

lemma (*in encoding*) $symm\text{-}closure\text{-}of\text{-}indRelLT$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $refl: refl\ TRel$
and $symm: sym\ TRel$
shows $indRelT\ TRel = symcl\ (indRelLT\ TRel)$
and $indRelTEQ\ TRel = (symcl\ ((indRelLT\ TRel)=))^+$
 $\langle proof \rangle$

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

lemma (*in encoding*) $indRelRT\text{-}to\text{-}TRel$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $TP\ TQ :: 'procT$
assumes $rel: TargetTerm\ TP\ \mathcal{R}[\cdot]RT < TRel > TargetTerm\ TQ$
shows $(TP, TQ) \in TRel$
 $\langle proof \rangle$

lemma (*in encoding*) $indRelLT\text{-}to\text{-}TRel$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $TP\ TQ :: 'procT$
assumes $rel: TargetTerm\ TP\ \mathcal{R}[\cdot]LT < TRel > TargetTerm\ TQ$
shows $(TP, TQ) \in TRel$
 $\langle proof \rangle$

lemma (*in encoding*) $indRelT\text{-}to\text{-}TRel$:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $TP\ TQ :: 'procT$
assumes $rel: TargetTerm\ TP\ \mathcal{R}[\cdot]T < TRel > TargetTerm\ TQ$

shows $(TP, TQ) \in TRel$
 ⟨proof⟩

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

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $P Q :: ('procS, 'procT) \text{ Proc}$
assumes $rel: P \lesssim[\cdot] RT < TRel > Q$
shows $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow SP = SQ$
and $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
and $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow \text{False}$
and $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$
 ⟨proof⟩

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $P Q :: ('procS, 'procT) \text{ Proc}$
assumes $rel: P \lesssim[\cdot] LT < TRel > Q$
shows $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow SP = SQ$
and $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow \text{False}$
and $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
and $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$
 ⟨proof⟩

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $P Q :: ('procS, 'procT) \text{ Proc}$
assumes $rel: P \sim[\cdot] T < TRel > Q$
shows $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
and $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
and $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow (TP, \llbracket SQ \rrbracket) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
and $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
 ⟨proof⟩

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $TP TQ :: 'procT$
assumes $(TP, TQ) \in (TRel \cup \{(T1, T2). \exists S. T1 = \llbracket S \rrbracket \wedge T2 = \llbracket S \rrbracket\})^+$
shows $(TP, TQ) \in TRel^*$
 ⟨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 $TRel :: ('procT \times 'procT) \text{ set}$
and $S :: 'procS$
and $T :: 'procT$
assumes $pair: \text{SourceTerm } S \lesssim[\cdot] RT < TRel > \text{TargetTerm } T$
shows $(\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in (\text{indRelRTPO } TRel)^=$
 ⟨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 $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *preservation: rel-preserves-barbs Rel (STCalWB SWB TWB)*
and *targetInRel: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*
shows *rel-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelRTPO-impl-TRel-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelRTPO TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelLTPO-impl-TRel-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelLTPO TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelTEQ-impl-TRel-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelTEQ TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *rel-with-target-impl-TRel-weakly-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)*
and *targetInRel: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*
shows *rel-weakly-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-weakly-preserves-barbs (indRelRTPO TRel) (STCalWB SWB TWB)*
shows *rel-weakly-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelLTPO-impl-TRel-weakly-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-weakly-preserves-barbs (indRelLTPO TRel) (STCalWB SWB TWB)*
shows *rel-weakly-preserves-barbs TRel TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-preserves-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-weakly-preserves-barbs (indRelTEQ TRel) (STCalWB SWB TWB)*
shows *rel-weakly-preserves-barbs TRel TWB*
 $\langle proof \rangle$

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

lemma (*in encoding-wrt-barbs*) *rel-with-target-impl-TRel-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)*
and *targetInRel: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*

shows *rel-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-reflects-barbs* (*indRelRTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-reflects-barbs* (*indRelLTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-reflects-barbs* (*indRelTEQ* $TRel$) (*STCalWB SWB TWB*)
shows *rel-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *rel-with-target-impl-TRel-weakly-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$ *set*
assumes *reflection: rel-weakly-reflects-barbs* Rel (*STCalWB SWB TWB*)
and *targetInRel: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*
shows *rel-weakly-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-weakly-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-weakly-reflects-barbs* (*indRelRTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-weakly-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-weakly-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-weakly-reflects-barbs* (*indRelLTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-weakly-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-reflects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *reflection: rel-weakly-reflects-barbs* (*indRelTEQ* $TRel$) (*STCalWB SWB TWB*)
shows *rel-weakly-reflects-barbs* $TRel$ TWB
 ⟨*proof*⟩

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

lemma (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-respects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *respection: rel-respects-barbs* (*indRelRTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-respects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-respects-barbs*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *respection: rel-respects-barbs* (*indRelLTPO* $TRel$) (*STCalWB SWB TWB*)
shows *rel-respects-barbs* $TRel$ TWB
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-respects-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
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 $TRel :: ('procT \times 'procT) \text{ set}$
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 $TRel :: ('procT \times 'procT) \text{ set}$
assumes *respection: rel-weakly-respects-barbs* (*indRelLTPO TRel*) (*STCalWB SWB TWB*)
shows *rel-weakly-respects-barbs TRel TWB*
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-weakly-respects-barbs*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *respection: rel-weakly-respects-barbs* (*indRelTEQ TRel*) (*STCalWB SWB TWB*)
shows *rel-weakly-respects-barbs TRel TWB*
 ⟨*proof*⟩

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

lemma (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$
assumes *sim: weak-reduction-simulation Rel* (*STCal Source Target*)
and *target: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*
and *trel: $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$*
shows *weak-reduction-simulation (TRel⁺) Target*
 ⟨*proof*⟩

lemma (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *sim: weak-reduction-simulation (indRelRTPO TRel)* (*STCal Source Target*)
shows *weak-reduction-simulation (TRel⁺) Target*
 ⟨*proof*⟩

lemma (in *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *sim: weak-reduction-simulation (indRelLTPO TRel)* (*STCal Source Target*)
shows *weak-reduction-simulation (TRel⁺) Target*
 ⟨*proof*⟩

lemma (in *encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$
assumes *sim: weak-reduction-simulation (Rel⁻¹)* (*STCal Source Target*)
and *target: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$*
and *trel: $\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$*
shows *weak-reduction-simulation ((TRel⁺)⁻¹) Target*
 ⟨*proof*⟩

lemma (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *sim: weak-reduction-simulation ((indRelRTPO TRel)⁻¹)* (*STCal Source Target*)

shows *weak-reduction-simulation* $((TRel^+)^{-1})$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *sim*: *weak-reduction-simulation* $((indRelLTPO\ TRel)^{-1})$ $(STCal\ Source\ Target)$
shows *weak-reduction-simulation* $((TRel^+)^{-1})$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ *set*
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ *set*
assumes *sim*: *weak-reduction-simulation* Rel $(STCal\ Source\ Target)$
and *target*: $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and *trel*: $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$
shows *weak-reduction-simulation* $(TRel^*)$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *sim*: *weak-reduction-simulation* $(indRelTEQ\ TRel)$ $(STCal\ Source\ Target)$
shows *weak-reduction-simulation* $(TRel^*)$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ *set*
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ *set*
assumes *sim*: *strong-reduction-simulation* Rel $(STCal\ Source\ Target)$
and *target*: $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and *trel*: $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$
shows *strong-reduction-simulation* $(TRel^+)$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *sim*: *strong-reduction-simulation* $(indRelRTPO\ TRel)$ $(STCal\ Source\ Target)$
shows *strong-reduction-simulation* $(TRel^+)$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *indRelLTPO-impl-TRel-is-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *sim*: *strong-reduction-simulation* $(indRelLTPO\ TRel)$ $(STCal\ Source\ Target)$
shows *strong-reduction-simulation* $(TRel^+)$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev*:
fixes $TRel :: ('procT \times 'procT)$ *set*
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ *set*
assumes *sim*: *strong-reduction-simulation* (Rel^{-1}) $(STCal\ Source\ Target)$
and *target*: $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and *trel*: $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$
shows *strong-reduction-simulation* $((TRel^+)^{-1})$ *Target*
 ⟨*proof*⟩

lemma (*in encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev*:
fixes $TRel :: ('procT \times 'procT)$ *set*
assumes *sim*: *strong-reduction-simulation* $((indRelRTPO\ TRel)^{-1})$ $(STCal\ Source\ Target)$
shows *strong-reduction-simulation* $((TRel^+)^{-1})$ *Target*
 ⟨*proof*⟩

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

fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : strong-reduction-simulation $((indRelLTPO\ TRel)^{-1})$ (STCal Source Target)
shows strong-reduction-simulation $((TRel^+)^{-1})$ Target
 ⟨proof⟩

lemma (in encoding) *rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)$ set
assumes sim : strong-reduction-simulation Rel (STCal Source Target)
and $target$: $\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and $trel$: $\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
 $\longrightarrow (T1, T2) \in TRel^*$
shows strong-reduction-simulation $(TRel^*)$ Target
 ⟨proof⟩

lemma (in encoding) *indRelTEQ-impl-TRel-is-strong-reduction-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : strong-reduction-simulation $(indRelTEQ\ TRel)$ (STCal Source Target)
shows strong-reduction-simulation $(TRel^*)$ Target
 ⟨proof⟩

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

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

lemma (in encoding-wrt-barbs) *indRelTEQ-impl-TRel-is-weak-barbed-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : weak-barbed-simulation $(indRelTEQ\ TRel)$ (STCalWB SWB TWB)
shows weak-barbed-simulation $(TRel^*)$ TWB
 ⟨proof⟩

lemma (in encoding-wrt-barbs) *indRelRTPO-impl-TRel-is-strong-barbed-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : strong-barbed-simulation $(indRelRTPO\ TRel)$ (STCalWB SWB TWB)
shows strong-barbed-simulation $(TRel^+)$ TWB
 ⟨proof⟩

lemma (in encoding-wrt-barbs) *indRelLTPO-impl-TRel-is-strong-barbed-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : strong-barbed-simulation $(indRelLTPO\ TRel)$ (STCalWB SWB TWB)
shows strong-barbed-simulation $(TRel^+)$ TWB
 ⟨proof⟩

lemma (in encoding-wrt-barbs) *indRelTEQ-impl-TRel-is-strong-barbed-simulation*:
fixes $TRel :: ('procT \times 'procT)$ set
assumes sim : strong-barbed-simulation $(indRelTEQ\ TRel)$ (STCalWB SWB TWB)
shows strong-barbed-simulation $(TRel^*)$ TWB
 ⟨proof⟩

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

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

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

lemma (**in encoding**) $\text{indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-reduction-contrasimulation } (indRelRTPO TRel) \text{ (STCal Source Target)}$
shows $\text{weak-reduction-contrasimulation } (TRel^+) \text{ Target}$
 $\langle proof \rangle$

lemma (**in encoding**) $\text{indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-reduction-contrasimulation } (indRelLTPO TRel) \text{ (STCal Source Target)}$
shows $\text{weak-reduction-contrasimulation } (TRel^+) \text{ Target}$
 $\langle proof \rangle$

lemma (**in encoding**) $\text{rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes $conSim: \text{weak-reduction-contrasimulation } Rel \text{ (STCal Source Target)}$
and $target: \forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$
and $trel: \forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$
shows $\text{weak-reduction-contrasimulation } (TRel^*) \text{ Target}$
 $\langle proof \rangle$

lemma (**in encoding**) $\text{indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-reduction-contrasimulation } (indRelTEQ TRel) \text{ (STCal Source Target)}$
shows $\text{weak-reduction-contrasimulation } (TRel^*) \text{ Target}$
 $\langle proof \rangle$

lemma (**in encoding-wrt-barbs**) $\text{indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-barbed-contrasimulation } (indRelRTPO TRel) \text{ (STCalWB SWB TWB)}$
shows $\text{weak-barbed-contrasimulation } (TRel^+) \text{ TWB}$
 $\langle proof \rangle$

lemma (**in encoding-wrt-barbs**) $\text{indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-barbed-contrasimulation } (indRelLTPO TRel) \text{ (STCalWB SWB TWB)}$
shows $\text{weak-barbed-contrasimulation } (TRel^+) \text{ TWB}$
 $\langle proof \rangle$

lemma (**in encoding-wrt-barbs**) $\text{indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $conSim: \text{weak-barbed-contrasimulation } (indRelTEQ TRel) \text{ (STCalWB SWB TWB)}$
shows $\text{weak-barbed-contrasimulation } (TRel^*) \text{ TWB}$
 $\langle proof \rangle$

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

lemma (**in encoding**) $\text{indRelRTPO-impl-TRel-is-weak-reduction-coupled-simulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $couSim: \text{weak-reduction-coupled-simulation } (indRelRTPO TRel) \text{ (STCal Source Target)}$
shows $\text{weak-reduction-coupled-simulation } (TRel^+) \text{ Target}$
 $\langle proof \rangle$

lemma (in *encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *couSim*: *weak-reduction-coupled-simulation* (*indRelLTPO* $TRel$) (*STCal* *Source* *Target*)
shows *weak-reduction-coupled-simulation* ($TRel^+$) *Target*
 <proof>

lemma (in *encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-coupled-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *couSim*: *weak-reduction-coupled-simulation* (*indRelTEQ* $TRel$) (*STCal* *Source* *Target*)
shows *weak-reduction-coupled-simulation* ($TRel^*$) *Target*
 <proof>

lemma (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-coupled-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *couSim*: *weak-barbed-coupled-simulation* (*indRelRTPO* $TRel$) (*STCalWB* *SWB* *TWB*)
shows *weak-barbed-coupled-simulation* ($TRel^+$) *TWB*
 <proof>

lemma (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-coupled-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *couSim*: *weak-barbed-coupled-simulation* (*indRelLTPO* $TRel$) (*STCalWB* *SWB* *TWB*)
shows *weak-barbed-coupled-simulation* ($TRel^+$) *TWB*
 <proof>

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

lemma (in *encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
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 $TRel :: ('procT \times 'procT) \text{ set}$
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-reflC-transC-TRel-is-weak-reduction-correspondence-simulation:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$
assumes *corSim*: *weak-reduction-correspondence-simulation* Rel (*STCal* *Source* *Target*)
and *target*: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel$

and $trel: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$
shows *weak-reduction-correspondence-simulation* ($TRel^*$) *Target*
 $\langle proof \rangle$

lemma (*in encoding*) *indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $corSim: weak-reduction-correspondence-simulation\ (indRelTEQ\ TRel)\ (STCal\ Source\ Target)$
shows *weak-reduction-correspondence-simulation* ($TRel^*$) *Target*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-correspondence-simulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $corSim: weak-barbed-correspondence-simulation\ (indRelRTPO\ TRel)\ (STCalWB\ SWB\ TWB)$
shows *weak-barbed-correspondence-simulation* ($TRel^+$) *TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-correspondence-simulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $corSim: weak-barbed-correspondence-simulation\ (indRelLTPO\ TRel)\ (STCalWB\ SWB\ TWB)$
shows *weak-barbed-correspondence-simulation* ($TRel^+$) *TWB*
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-correspondence-simulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $corSim: weak-barbed-correspondence-simulation\ (indRelTEQ\ TRel)\ (STCalWB\ SWB\ TWB)$
shows *weak-barbed-correspondence-simulation* ($TRel^*$) *TWB*
 $\langle proof \rangle$

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

lemma (*in encoding*) *rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$
assumes $bisim: weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$
and $target: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and $trel: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+$
shows *weak-reduction-bisimulation* ($TRel^+$) *Target*
 $\langle proof \rangle$

lemma (*in encoding*) *indRelRTPO-impl-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $bisim: weak-reduction-bisimulation\ (indRelRTPO\ TRel)\ (STCal\ Source\ Target)$
shows *weak-reduction-bisimulation* ($TRel^+$) *Target*
 $\langle proof \rangle$

lemma (*in encoding*) *indRelLTPO-impl-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $bisim: weak-reduction-bisimulation\ (indRelLTPO\ TRel)\ (STCal\ Source\ Target)$
shows *weak-reduction-bisimulation* ($TRel^+$) *Target*
 $\langle proof \rangle$

lemma (*in encoding*) *rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT)\ set$
and $Rel :: (('procS, 'procT)\ Proc \times ('procS, 'procT)\ Proc)\ set$
assumes $bisim: weak-reduction-bisimulation\ Rel\ (STCal\ Source\ Target)$
and $target: \forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel$
and $trel: \forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^*$
shows *weak-reduction-bisimulation* ($TRel^*$) *Target*
 $\langle proof \rangle$

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

lemma (in *encoding*) *indRelRTPO-impl-TRel-is-strong-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
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 $TRel :: ('procT \times 'procT) \text{ set}$
assumes *bisim: strong-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)*
shows *strong-reduction-bisimulation (TRel+) Target*
 ⟨*proof*⟩

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

lemma (in *encoding*) *indRelTEQ-impl-TRel-is-strong-reduction-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *bisim: strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)*
shows *strong-reduction-bisimulation (TRel*) Target*
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelRTPO-impl-TRel-is-weak-barbed-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *bisim: weak-barbed-bisimulation (indRelRTPO TRel) (STCalWB SWB TWB)*
shows *weak-barbed-bisimulation (TRel+) TWB*
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelLTPO-impl-TRel-is-weak-barbed-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *bisim: weak-barbed-bisimulation (indRelLTPO TRel) (STCalWB SWB TWB)*
shows *weak-barbed-bisimulation (TRel+) TWB*
 ⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelTEQ-impl-TRel-is-weak-barbed-bisimulation*:
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes *bisim: weak-barbed-bisimulation (indRelTEQ TRel) (STCalWB SWB TWB)*
shows *weak-barbed-bisimulation (TRel*) TWB*
 ⟨*proof*⟩

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $bisim: \text{strong-barbed-bisimulation } (indRelRTPO \ TRel) \ (STCalWB \ SWB \ TWB)$
shows $\text{strong-barbed-bisimulation } (TRel^+) \ TWB$
 $\langle \text{proof} \rangle$

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

lemma (**in** $\text{encoding-wrt-barbs}$) $indRelTEQ\text{-impl-}TRel\text{-is-strong-barbed-bisimulation:}$
fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes $bisim: \text{strong-barbed-bisimulation } (indRelTEQ \ TRel) \ (STCalWB \ SWB \ TWB)$
shows $\text{strong-barbed-bisimulation } (TRel^*) \ TWB$
 $\langle \text{proof} \rangle$

5.3 Relations Induced by the Encoding and Relations on Source Terms and Target Terms

Some encodability like e.g. full abstraction are defined w.r.t. a relation on source terms and a relation on target terms. To analyse such criteria we include these two relations in the considered relation on the disjoint union of source and target terms.

inductive-set (**in** encoding) $indRelRST$
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow (((('procS, 'procT) \ Proc) \times (('procS, 'procT) \ Proc)) \ \text{set})$
for $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
where
 $encR: (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in indRelRST \ SRel \ TRel \ |$
 $source: (S1, S2) \in SRel \Longrightarrow (SourceTerm \ S1, SourceTerm \ S2) \in indRelRST \ SRel \ TRel \ |$
 $target: (T1, T2) \in TRel \Longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in indRelRST \ SRel \ TRel$

abbreviation (**in** encoding) $indRelRSTinfix$
 $:: ('procS, 'procT) \ Proc \Rightarrow ('procS \times 'procS) \ \text{set} \Rightarrow ('procT \times 'procT) \ \text{set}$
 $\Rightarrow ('procS, 'procT) \ Proc \Rightarrow \text{bool } (- \ \mathcal{R}[\cdot]R<-,> \ - \ [75, 75, 75, 75] \ 80)$
where
 $P \ \mathcal{R}[\cdot]R<SRel, TRel> \ Q \equiv (P, Q) \in indRelRST \ SRel \ TRel$

inductive-set (**in** encoding) $indRelRSTPO$
 $:: ('procS \times 'procS) \ \text{set} \Rightarrow ('procT \times 'procT) \ \text{set}$
 $\Rightarrow (((('procS, 'procT) \ Proc) \times (('procS, 'procT) \ Proc)) \ \text{set})$
for $SRel :: ('procS \times 'procS) \ \text{set}$
and $TRel :: ('procT \times 'procT) \ \text{set}$
where
 $encR: (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in indRelRSTPO \ SRel \ TRel \ |$
 $source: (S1, S2) \in SRel \Longrightarrow (SourceTerm \ S1, SourceTerm \ S2) \in indRelRSTPO \ SRel \ TRel \ |$
 $target: (T1, T2) \in TRel \Longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in indRelRSTPO \ SRel \ TRel \ |$
 $trans: \llbracket (P, Q) \in indRelRSTPO \ SRel \ TRel; (Q, R) \in indRelRSTPO \ SRel \ TRel \rrbracket$
 $\Longrightarrow (P, R) \in indRelRSTPO \ SRel \ TRel$

abbreviation (**in** encoding) $indRelRSTPOinfix ::$
 $(('procS, 'procT) \ Proc) \Rightarrow ('procS \times 'procS) \ \text{set} \Rightarrow ('procT \times 'procT) \ \text{set}$
 $\Rightarrow ('procS, 'procT) \ Proc \Rightarrow \text{bool } (- \ \lesssim[\cdot]R<-,> \ - \ [75, 75, 75, 75] \ 80)$
where
 $P \ \lesssim[\cdot]R<SRel, TRel> \ Q \equiv (P, Q) \in indRelRSTPO \ SRel \ TRel$

lemma (**in** encoding) $indRelRSTPO\text{-refl:}$
fixes $SRel :: ('procS \times 'procS) \ \text{set}$
and $TRel :: ('procT \times 'procT) \ \text{set}$

assumes $\text{reflS}: \text{refl } S\text{Rel}$
and $\text{reflT}: \text{refl } T\text{Rel}$
shows $\text{refl } (\text{indRelRSTPO } S\text{Rel } T\text{Rel})$
 $\langle \text{proof} \rangle$

lemma (**in encoding**) indRelRSTPO-trans :
fixes $S\text{Rel} :: ('procS \times 'procS) \text{ set}$
and $T\text{Rel} :: ('procT \times 'procT) \text{ set}$
shows $\text{trans } (\text{indRelRSTPO } S\text{Rel } T\text{Rel})$
 $\langle \text{proof} \rangle$

lemma (**in encoding**) $\text{refl-trans-closure-of-indRelRST}$:
fixes $S\text{Rel} :: ('procS \times 'procS) \text{ set}$
and $T\text{Rel} :: ('procT \times 'procT) \text{ set}$
assumes $\text{reflS}: \text{refl } S\text{Rel}$
and $\text{reflT}: \text{refl } T\text{Rel}$
shows $\text{indRelRSTPO } S\text{Rel } T\text{Rel} = (\text{indRelRST } S\text{Rel } T\text{Rel})^*$
 $\langle \text{proof} \rangle$

inductive-set (**in encoding**) indRelLST
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow (((('procS, 'procT) \text{ Proc}) \times ((('procS, 'procT) \text{ Proc}))) \text{ set}$
for $S\text{Rel} :: ('procS \times 'procS) \text{ set}$
and $T\text{Rel} :: ('procT \times 'procT) \text{ set}$
where
 $\text{encL}: (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{indRelLST } S\text{Rel } T\text{Rel} \mid$
 $\text{source}: (S1, S2) \in S\text{Rel} \Longrightarrow (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{indRelLST } S\text{Rel } T\text{Rel} \mid$
 $\text{target}: (T1, T2) \in T\text{Rel} \Longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelLST } S\text{Rel } T\text{Rel}$

abbreviation (**in encoding**) indRelLSTinfix
 $:: ('procS, 'procT) \text{ Proc} \Rightarrow ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool } (- \mathcal{R}[\cdot]L<-,> - [75, 75, 75, 75] 80)$
where
 $P \mathcal{R}[\cdot]L<S\text{Rel}, T\text{Rel}> Q \equiv (P, Q) \in \text{indRelLST } S\text{Rel } T\text{Rel}$

inductive-set (**in encoding**) indRelLSTPO
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow (((('procS, 'procT) \text{ Proc}) \times ((('procS, 'procT) \text{ Proc}))) \text{ set}$
for $S\text{Rel} :: ('procS \times 'procS) \text{ set}$
and $T\text{Rel} :: ('procT \times 'procT) \text{ set}$
where
 $\text{encL}: (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel} \mid$
 $\text{source}: (S1, S2) \in S\text{Rel} \Longrightarrow (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel} \mid$
 $\text{target}: (T1, T2) \in T\text{Rel} \Longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel} \mid$
 $\text{trans}: \llbracket (P, Q) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel}; (Q, R) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel} \rrbracket$
 $\Longrightarrow (P, R) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel}$

abbreviation (**in encoding**) indRelLSTPOinfix
 $:: ('procS, 'procT) \text{ Proc} \Rightarrow ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool } (- \lesssim[\cdot]L<-,> - [75, 75, 75, 75] 80)$
where
 $P \lesssim[\cdot]L<S\text{Rel}, T\text{Rel}> Q \equiv (P, Q) \in \text{indRelLSTPO } S\text{Rel } T\text{Rel}$

lemma (**in encoding**) indRelLSTPO-refl :
fixes $S\text{Rel} :: ('procS \times 'procS) \text{ set}$
and $T\text{Rel} :: ('procT \times 'procT) \text{ set}$
assumes $\text{reflS}: \text{refl } S\text{Rel}$
and $\text{reflT}: \text{refl } T\text{Rel}$
shows $\text{refl } (\text{indRelLSTPO } S\text{Rel } T\text{Rel})$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelLSTPO-trans*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
shows $\text{trans } (indRelLSTPO \ SRel \ TRel)$
<proof>

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

inductive-set (in *encoding*) *indRelST*
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow (((('procS, 'procT) \ Proc) \times (('procS, 'procT) \ Proc)) \text{ set})$
for $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
where
 $\text{encR: } (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in indRelST \ SRel \ TRel \ |$
 $\text{encL: } (TargetTerm \ (\llbracket S \rrbracket), SourceTerm \ S) \in indRelST \ SRel \ TRel \ |$
 $\text{source: } (S1, S2) \in SRel \Longrightarrow (SourceTerm \ S1, SourceTerm \ S2) \in indRelST \ SRel \ TRel \ |$
 $\text{target: } (T1, T2) \in TRel \Longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in indRelST \ SRel \ TRel$

abbreviation (in *encoding*) *indRelSTinfix*
 $:: ('procS, 'procT) \ Proc \Rightarrow ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow ('procS, 'procT) \ Proc \Rightarrow \text{bool } (- \ \mathcal{R}[\cdot] \langle -, - \rangle - \ [75, 75, 75, 75] \ 80)$
where
 $P \ \mathcal{R}[\cdot] \langle SRel, TRel \rangle \ Q \equiv (P, Q) \in indRelST \ SRel \ TRel$

lemma (in *encoding*) *indRelST-symm*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes $\text{symmS: sym } SRel$
and $\text{symmT: sym } TRel$
shows $\text{sym } (indRelST \ SRel \ TRel)$
<proof>

inductive-set (in *encoding*) *indRelSTEQ*
 $:: ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow (((('procS, 'procT) \ Proc) \times (('procS, 'procT) \ Proc)) \text{ set})$
for $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
where
 $\text{encR: } (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in indRelSTEQ \ SRel \ TRel \ |$
 $\text{encL: } (TargetTerm \ (\llbracket S \rrbracket), SourceTerm \ S) \in indRelSTEQ \ SRel \ TRel \ |$
 $\text{source: } (S1, S2) \in SRel \Longrightarrow (SourceTerm \ S1, SourceTerm \ S2) \in indRelSTEQ \ SRel \ TRel \ |$
 $\text{target: } (T1, T2) \in TRel \Longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in indRelSTEQ \ SRel \ TRel \ |$
 $\text{trans: } \llbracket (P, Q) \in indRelSTEQ \ SRel \ TRel; (Q, R) \in indRelSTEQ \ SRel \ TRel \rrbracket$
 $\Longrightarrow (P, R) \in indRelSTEQ \ SRel \ TRel$

abbreviation (in *encoding*) *indRelSTEQinfix*
 $:: ('procS, 'procT) \ Proc \Rightarrow ('procS \times 'procS) \text{ set} \Rightarrow ('procT \times 'procT) \text{ set}$
 $\Rightarrow ('procS, 'procT) \ Proc \Rightarrow \text{bool } (- \ \sim[\cdot] \langle -, - \rangle - \ [75, 75, 75, 75] \ 80)$
where
 $P \ \sim[\cdot] \langle SRel, TRel \rangle \ Q \equiv (P, Q) \in indRelSTEQ \ SRel \ TRel$

lemma (in *encoding*) *indRelSTEQ-refl*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$

assumes $reflT: refl\ TRel$
shows $refl\ (indRelSTEQ\ SRel\ TRel)$
 $\langle proof \rangle$

lemma (in *encoding*) $indRelSTEQ\text{-}symm$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $symmS: sym\ SRel$
and $symmT: sym\ TRel$
shows $sym\ (indRelSTEQ\ SRel\ TRel)$
 $\langle proof \rangle$

lemma (in *encoding*) $indRelSTEQ\text{-}trans$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
shows $trans\ (indRelSTEQ\ SRel\ TRel)$
 $\langle proof \rangle$

lemma (in *encoding*) $refl\text{-}trans\text{-}closure\text{-}of\text{-}indRelST$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $reflT: refl\ TRel$
shows $indRelSTEQ\ SRel\ TRel = (indRelST\ SRel\ TRel)^*$
 $\langle proof \rangle$

lemma (in *encoding*) $refl\text{-}symm\text{-}trans\text{-}closure\text{-}of\text{-}indRelST$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $reflT: refl\ TRel$
and $symmS: sym\ SRel$
and $symmT: sym\ TRel$
shows $indRelSTEQ\ SRel\ TRel = (symcl\ ((indRelST\ SRel\ TRel)^=))^+$
 $\langle proof \rangle$

lemma (in *encoding*) $symm\text{-}closure\text{-}of\text{-}indRelRST$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $reflT: refl\ TRel$
and $symmS: sym\ SRel$
and $symmT: sym\ TRel$
shows $indRelRST\ SRel\ TRel = symcl\ (indRelRST\ SRel\ TRel)$
and $indRelSTEQ\ SRel\ TRel = (symcl\ ((indRelRST\ SRel\ TRel)^=))^+$
 $\langle proof \rangle$

lemma (in *encoding*) $symm\text{-}closure\text{-}of\text{-}indRelLST$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $reflT: refl\ TRel$
and $symmS: sym\ SRel$
and $symmT: sym\ TRel$
shows $indRelLST\ SRel\ TRel = symcl\ (indRelLST\ SRel\ TRel)$
and $indRelSTEQ\ SRel\ TRel = (symcl\ ((indRelLST\ SRel\ TRel)^=))^+$
 $\langle proof \rangle$

lemma (in *encoding*) $symm\text{-}trans\text{-}closure\text{-}of\text{-}indRelRSTPO$:
fixes $SRel :: ('procS \times 'procS)\ set$
and $TRel :: ('procT \times 'procT)\ set$
assumes $symmS: sym\ SRel$
and $symmT: sym\ TRel$
shows $indRelRSTPO\ SRel\ TRel = (symcl\ (indRelRSTPO\ SRel\ TRel))^+$
 $\langle proof \rangle$

lemma (in *encoding*) *symm-trans-closure-of-indRelLSTPO*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes $symmS: \text{sym } SRel$
and $symmT: \text{sym } TRel$
shows $\text{indRelSTEQ } SRel \ TRel = (\text{symcl } (\text{indRelLSTPO } SRel \ TRel))^+$
 $\langle \text{proof} \rangle$

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 $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $SP \ SQ :: 'procS$
assumes $\text{rel}: \text{SourceTerm } SP \ \mathcal{R}[\cdot]R \langle SRel, TRel \rangle \ \text{SourceTerm } SQ$
shows $(SP, SQ) \in SRel$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelRST-to-TRel*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $TP \ TQ :: 'procT$
assumes $\text{rel}: \text{TargetTerm } TP \ \mathcal{R}[\cdot]R \langle SRel, TRel \rangle \ \text{TargetTerm } TQ$
shows $(TP, TQ) \in TRel$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelLST-to-SRel*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $SP \ SQ :: 'procS$
assumes $\text{rel}: \text{SourceTerm } SP \ \mathcal{R}[\cdot]L \langle SRel, TRel \rangle \ \text{SourceTerm } SQ$
shows $(SP, SQ) \in SRel$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelLST-to-TRel*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $TP \ TQ :: 'procT$
assumes $\text{rel}: \text{TargetTerm } TP \ \mathcal{R}[\cdot]L \langle SRel, TRel \rangle \ \text{TargetTerm } TQ$
shows $(TP, TQ) \in TRel$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelST-to-SRel*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $SP \ SQ :: 'procS$
assumes $\text{rel}: \text{SourceTerm } SP \ \mathcal{R}[\cdot] \langle SRel, TRel \rangle \ \text{SourceTerm } SQ$
shows $(SP, SQ) \in SRel$
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *indRelST-to-TRel*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $TP \ TQ :: 'procT$
assumes $\text{rel}: \text{TargetTerm } TP \ \mathcal{R}[\cdot] \langle SRel, TRel \rangle \ \text{TargetTerm } TQ$
shows $(TP, TQ) \in TRel$
 $\langle \text{proof} \rangle$

If the relations indRelRSTPO or indRelLSTPO contain a pair of target terms, then this pair is also

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

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $P Q :: ('procS, 'procT) \text{ Proc}$
assumes $P \lesssim_{\llbracket \cdot \rrbracket R} \langle SRel, TRel \rangle Q$
shows $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+$
and $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow (\exists S. (SP, S) \in SRel^* \wedge (\llbracket S \rrbracket, TQ) \in TRel^*)$
and $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow \text{False}$
and $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$
 $\langle \text{proof} \rangle$

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $P Q :: ('procS, 'procT) \text{ Proc}$
assumes $P \lesssim_{\llbracket \cdot \rrbracket L} \langle SRel, TRel \rangle Q$
shows $\forall SP SQ. SP \in S P \wedge SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+$
and $\forall SP TQ. SP \in S P \wedge TQ \in T Q \longrightarrow \text{False}$
and $\forall TP SQ. TP \in T P \wedge SQ \in S Q \longrightarrow (\exists S. (TP, \llbracket S \rrbracket) \in TRel^* \wedge (S, SQ) \in SRel^*)$
and $\forall TP TQ. TP \in T P \wedge TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+$
 $\langle \text{proof} \rangle$

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 $SRel :: ('procS \times 'procS) \text{ set}$
and $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$
assumes *preservation: rel-preserves-barbs Rel (STCalWB SWB TWB)*
and *sourceInRel: $\forall S1 S2. (S1, S2) \in SRel \longrightarrow (\text{SourceTerm } S1, \text{SourceTerm } S2) \in Rel$*
shows *rel-preserves-barbs SRel SWB*
 $\langle \text{proof} \rangle$

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs SRel SWB*
and *rel-preserves-barbs TRel TWB*
 $\langle \text{proof} \rangle$

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs SRel SWB*
and *rel-preserves-barbs TRel TWB*
 $\langle \text{proof} \rangle$

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *preservation: rel-preserves-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)*
shows *rel-preserves-barbs SRel SWB*
and *rel-preserves-barbs TRel TWB*
 $\langle \text{proof} \rangle$

lemma (in *encoding-wrt-barbs*) *rel-with-source-impl-SRel-weakly-preserves-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)*
and *sourceInRel: $\forall S1 S2. (S1, S2) \in SRel \longrightarrow (SourceTerm S1, SourceTerm S2) \in Rel$*
shows *rel-weakly-preserves-barbs SRel SWB*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
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*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
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*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
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*
 $\langle proof \rangle$

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 $SRel :: ('procS \times 'procS) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)*
and *sourceInRel: $\forall S1 S2. (S1, S2) \in SRel \longrightarrow (SourceTerm S1, SourceTerm S2) \in Rel$*
shows *rel-reflects-barbs SRel SWB*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-reflects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)*
shows *rel-reflects-barbs SRel SWB*
and *rel-reflects-barbs TRel TWB*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-reflects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)*
shows *rel-reflects-barbs SRel SWB*
and *rel-reflects-barbs TRel TWB*
 $\langle proof \rangle$

lemma (in *encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$

and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-reflects-barbs* ($indRelSTEQ \ SRel \ TRel$) ($STCalWB \ SWB \ TWB$)
shows *rel-reflects-barbs* $SRel \ SWB$
and *rel-reflects-barbs* $TRel \ TWB$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *rel-with-source-impl-SRel-weakly-reflects-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $Rel :: (('procS, 'procT) \text{ Proc} \times ('procS, 'procT) \text{ Proc}) \text{ set}$
assumes *reflection: rel-weakly-reflects-barbs* Rel ($STCalWB \ SWB \ TWB$)
and *sourceInRel: $\forall S1 \ S2. (S1, S2) \in SRel \longrightarrow (SourceTerm \ S1, SourceTerm \ S2) \in Rel$*
shows *rel-weakly-reflects-barbs* $SRel \ SWB$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-weakly-reflects-barbs* ($indRelRSTPO \ SRel \ TRel$) ($STCalWB \ SWB \ TWB$)
shows *rel-weakly-reflects-barbs* $SRel \ SWB$
and *rel-weakly-reflects-barbs* $TRel \ TWB$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-weakly-reflects-barbs* ($indRelLSTPO \ SRel \ TRel$) ($STCalWB \ SWB \ TWB$)
shows *rel-weakly-reflects-barbs* $SRel \ SWB$
and *rel-weakly-reflects-barbs* $TRel \ TWB$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *reflection: rel-weakly-reflects-barbs* ($indRelSTEQ \ SRel \ TRel$) ($STCalWB \ SWB \ TWB$)
shows *rel-weakly-reflects-barbs* $SRel \ SWB$
and *rel-weakly-reflects-barbs* $TRel \ TWB$
 $\langle proof \rangle$

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

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

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

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

shows *rel-respects-barbs* *SRel SWB*
and *rel-respects-barbs* *TRel TWB*
⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelRSTPO-impl-SRel-and-TRel-weakly-respect-barbs*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
assumes *respection: rel-weakly-respects-barbs* (*indRelRSTPO SRel TRel*) (*STCalWB SWB TWB*)
shows *rel-weakly-respects-barbs* *SRel SWB*
and *rel-weakly-respects-barbs* *TRel TWB*
⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelLSTPO-impl-SRel-and-TRel-weakly-respect-barbs*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
assumes *respection: rel-weakly-respects-barbs* (*indRelLSTPO SRel TRel*) (*STCalWB SWB TWB*)
shows *rel-weakly-respects-barbs* *SRel SWB*
and *rel-weakly-respects-barbs* *TRel TWB*
⟨*proof*⟩

lemma (in *encoding-wrt-barbs*) *indRelSTEQ-impl-SRel-and-TRel-weakly-respect-barbs*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
assumes *respection: rel-weakly-respects-barbs* (*indRelSTEQ SRel TRel*) (*STCalWB SWB TWB*)
shows *rel-weakly-respects-barbs* *SRel SWB*
and *rel-weakly-respects-barbs* *TRel TWB*
⟨*proof*⟩

If *TRel* is reflexive then *ind relRTPO* is a subrelation of *indRelTEQ*. If *SRel* is reflexive then *indRelRTPO* is a subrelation of *indRelRTPO*. Moreover, *indRelRSTPO* is a subrelation of *indRelSTEQ*.

lemma (in *encoding*) *indRelRTPO-to-indRelTEQ*:
fixes *TRel* :: ('procT × 'procT) set
and *P Q* :: ('procS, 'procT) Proc
assumes *rel*: $P \lesssim[\cdot]RT \langle TRel \rangle Q$
and *reflT*: *refl TRel*
shows $P \sim[\cdot]T \langle TRel \rangle Q$
⟨*proof*⟩

lemma (in *encoding*) *indRelRTPO-to-indRelRSTPO*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
and *P Q* :: ('procS, 'procT) Proc
assumes *rel*: $P \lesssim[\cdot]RT \langle TRel \rangle Q$
and *reflS*: *refl SRel*
shows $P \lesssim[\cdot]R \langle SRel, TRel \rangle Q$
⟨*proof*⟩

lemma (in *encoding*) *indRelRSTPO-to-indRelSTEQ*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
and *P Q* :: ('procS, 'procT) Proc
assumes *rel*: $P \lesssim[\cdot]R \langle SRel, TRel \rangle Q$
shows $P \sim[\cdot] \langle SRel, TRel \rangle Q$
⟨*proof*⟩

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

lemma (in *encoding*) *indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation*:
fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set


```

assumes bisimT: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
and bisimS: weak-reduction-bisimulation SRel Source
and reflS: refl SRel
shows weak-reduction-bisimulation (indRelRSTPO SRel TRel) (STCal Source Target)
⟨proof⟩

end
theory SuccessSensitiveness
imports SourceTargetRelation
begin

```

6 Success Sensitiveness and Barbs

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

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

abbreviation (**in** *encoding-wrt-barbs*) *enc-weakly-preserves-barbs* :: bool **where**
enc-weakly-preserves-barbs \equiv *enc-preserves-binary-pred* ($\lambda P a. P \Downarrow .a$)

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

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

abbreviation (**in** *encoding-wrt-barbs*) *enc-weakly-reflects-barbs* :: bool **where**
enc-weakly-reflects-barbs \equiv *enc-reflects-binary-pred* ($\lambda P a. P \Downarrow .a$)

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

abbreviation (**in** *encoding-wrt-barbs*) *enc-weakly-respects-barb-set* :: 'barbs set \Rightarrow bool **where**
enc-weakly-respects-barb-set Barbs \equiv
enc-weakly-preserves-barb-set Barbs \wedge *enc-weakly-reflects-barb-set Barbs*

abbreviation (**in** *encoding-wrt-barbs*) *enc-weakly-respects-barbs* :: bool **where**
enc-weakly-respects-barbs \equiv *enc-weakly-preserves-barbs* \wedge *enc-weakly-reflects-barbs*

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

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

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

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

$enc\text{-preserves}\text{-barbs} \equiv enc\text{-preserves}\text{-binary}\text{-pred} (\lambda P a. P \downarrow . a)$

lemma (in *encoding-wrt-barbs*) *enc-preserves-barbs-and-barb-set*:
shows $enc\text{-preserves}\text{-barbs} = (\forall Barbs. enc\text{-preserves}\text{-barb}\text{-set} Barbs)$
 ⟨proof⟩

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

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

lemma (in *encoding-wrt-barbs*) *enc-reflects-barbs-and-barb-set*:
shows $enc\text{-reflects}\text{-barbs} = (\forall Barbs. enc\text{-reflects}\text{-barb}\text{-set} Barbs)$
 ⟨proof⟩

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

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

lemma (in *encoding-wrt-barbs*) *enc-respects-barbs-and-barb-set*:
shows $enc\text{-respects}\text{-barbs} = (\forall Barbs. enc\text{-respects}\text{-barb}\text{-set} Barbs)$
 ⟨proof⟩

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

lemma (in *encoding-wrt-barbs*) *enc-weakly-preserves-barb-set-iff-source-target-rel*:
fixes $Barbs :: 'barbs\ set$
and $TRel :: ('procT \times 'procT)\ set$
shows $enc\text{-weakly}\text{-preserves}\text{-barb}\text{-set} Barbs$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel\text{-weakly}\text{-preserves}\text{-barb}\text{-set} Rel (STCalWB\ SWB\ TWB)\ Barbs)$
 ⟨proof⟩

lemma (in *encoding-wrt-barbs*) *enc-weakly-preserves-barbs-iff-source-target-rel*:
shows $enc\text{-weakly}\text{-preserves}\text{-barbs}$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel\text{-weakly}\text{-preserves}\text{-barbs} Rel (STCalWB\ SWB\ TWB))$
 ⟨proof⟩

lemma (in *encoding-wrt-barbs*) *enc-preserves-barb-set-iff-source-target-rel*:
fixes $Barbs :: 'barbs\ set$
shows $enc\text{-preserves}\text{-barb}\text{-set} Barbs$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel\text{-preserves}\text{-barb}\text{-set} Rel (STCalWB\ SWB\ TWB)\ Barbs)$
 ⟨proof⟩

lemma (in *encoding-wrt-barbs*) *enc-preserves-barbs-iff-source-target-rel*:
shows $enc\text{-preserves}\text{-barbs}$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel\text{-preserves}\text{-barbs} Rel (STCalWB\ SWB\ TWB))$
 ⟨proof⟩

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

ity/existence of barbs.

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

fixes *Barbs* :: 'barbs set

shows *enc-weakly-reflects-barb-set Barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-reflects-barb-set Rel (STCalWB SWB TWB) Barbs})$$

<proof>

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

shows *enc-weakly-reflects-barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-reflects-barbs Rel (STCalWB SWB TWB)})$$

<proof>

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

fixes *Barbs* :: 'barbs set

shows *enc-reflects-barb-set Barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-reflects-barb-set Rel (STCalWB SWB TWB) Barbs})$$

<proof>

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

shows *enc-reflects-barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-reflects-barbs Rel (STCalWB SWB TWB)})$$

<proof>

An encoding (weakly) respects barbs iff (1) there exists a relation, like *indRelR*, that relates source terms and their literal translations and respects (reachability/existence of barbs), or (2) there exists a relation, like *indRelL*, that relates literal translations and their source terms and respects (reachability/existence of barbs), or (3) there exists a relation, like *indRel*, that relates source terms and their literal translations in both directions and respects (reachability/existence of barbs).

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

fixes *Barbs* :: 'barbs set

shows *enc-weakly-respects-barb-set Barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) Barbs})$$

<proof>

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

shows *enc-weakly-respects-barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-weakly-respects-barbs Rel (STCalWB SWB TWB)})$$

<proof>

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

fixes *Barbs* :: 'barbs set

shows *enc-respects-barb-set Barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-respects-barb-set Rel (STCalWB SWB TWB) Barbs})$$

<proof>

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

shows *enc-respects-barbs*

$$= (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}) \\ \wedge \text{rel-respects-barbs Rel (STCalWB SWB TWB)})$$

<proof>

Accordingly an encoding is success sensitive iff there exists such a relation between source and target terms that weakly respects the barb success.

lemma (in *encoding-wrt-barbs*) *success-sensitive-cond*:
fixes *success* :: 'barbs
shows *enc-weakly-respects-barb-set* {*success*} = $(\forall S. S \Downarrow \langle SWB \rangle \text{success} \longleftrightarrow \llbracket S \rrbracket \Downarrow \langle TWB \rangle \text{success})$
 ⟨*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*}
 = $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge \text{rel-weakly-respects-barb-set } \text{Rel } (\text{STCalWB } SWB \text{ TWB}) \{ \text{success} \})$
 ⟨*proof*⟩

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

end
theory *DivergenceReflection*
imports *SourceTargetRelation*
begin

7 Divergence Reflection

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

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

lemma (in *encoding*) *divergence-preservation-cond*:
shows *enc-preserves-divergence* = $(\forall S. S \mapsto (\text{Source})\omega \longrightarrow \llbracket S \rrbracket \mapsto (\text{Target})\omega)$
 ⟨*proof*⟩

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

lemma (in *encoding*) *divergence-reflection-cond*:
shows *enc-reflects-divergence* = $(\forall S. \llbracket S \rrbracket \mapsto (\text{Target})\omega \longrightarrow S \mapsto (\text{Source})\omega)$
 ⟨*proof*⟩

abbreviation *rel-preserves-divergence*
 :: ('proc × 'proc) set \Rightarrow 'proc processCalculus \Rightarrow bool
where
rel-preserves-divergence *Rel Cal* $\equiv \text{rel-preserves-pred } \text{Rel } (\lambda P. P \mapsto (\text{Cal})\omega)$

abbreviation *rel-reflects-divergence*
 :: ('proc × 'proc) set \Rightarrow 'proc processCalculus \Rightarrow bool
where
rel-reflects-divergence *Rel Cal* $\equiv \text{rel-reflects-pred } \text{Rel } (\lambda P. P \mapsto (\text{Cal})\omega)$

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

abbreviation (in *encoding*) *enc-respects-divergence* :: bool **where**
enc-respects-divergence $\equiv \text{enc-respects-pred } (\lambda P. P \mapsto \text{ST}\omega)$

lemma (in *encoding*) *divergence-respection-cond*:
shows *enc-respects-divergence* = $(\forall S. \llbracket S \rrbracket \mapsto (Target)\omega \longleftrightarrow S \mapsto (Source)\omega)$
 ⟨*proof*⟩

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

An encoding preserves divergence iff (1) there exists a relation that relates source terms and their literal translations and preserves divergence, or (2) there exists a relation that relates literal translations and their source terms and reflects divergence.

lemma (in *encoding*) *divergence-preservation-iff-source-target-rel-preserves-divergence*:
shows *enc-preserves-divergence*
 = $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-preserves-divergence\ Rel\ (STCal\ Source\ Target))$
 ⟨*proof*⟩

lemma (in *encoding*) *divergence-preservation-iff-source-target-rel-reflects-divergence*:
shows *enc-preserves-divergence*
 = $(\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target))$
 ⟨*proof*⟩

An encoding reflects divergence iff (1) there exists a relation that relates source terms and their literal translations and reflects divergence, or (2) there exists a relation that relates literal translations and their source terms and preserves divergence.

lemma (in *encoding*) *divergence-reflection-iff-source-target-rel-reflects-divergence*:
shows *enc-reflects-divergence*
 = $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target))$
 ⟨*proof*⟩

lemma (in *encoding*) *divergence-reflection-iff-source-target-rel-preserves-divergence*:
shows *enc-reflects-divergence*
 = $(\exists Rel. (\forall S. (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-preserves-divergence\ Rel\ (STCal\ Source\ Target))$
 ⟨*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* = $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-divergence\ Rel\ (STCal\ Source\ Target))$
and *enc-respects-divergence* = $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel) \wedge rel-respects-divergence\ Rel\ (STCal\ Source\ Target))$
 ⟨*proof*⟩

end
theory *OperationalCorrespondence*
imports *SourceTargetRelation*
begin

8 Operational Correspondence

We consider different variants of operational correspondence. This criterion consists of a completeness and a soundness condition and is often defined with respect to a relation TRel on target terms.

Operational completeness modulo TRel ensures that an encoding preserves source term behaviour modulo TRel by requiring that each sequence of source term steps can be mimicked by its translation such that the respective derivatives are related by TRel.

abbreviation (in *encoding*) *operational-complete* :: ('procT × 'procT) set ⇒ bool **where**
operational-complete TRel ≡
 $\forall S S'. S \mapsto \text{Source} * S' \longrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target} * T \wedge (\llbracket S \rrbracket, T) \in \text{TRel})$

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

abbreviation (in *encoding*) *strongly-operational-complete* :: ('procT × 'procT) set ⇒ bool **where**
strongly-operational-complete TRel ≡
 $\forall S S'. S \mapsto \text{Source} S' \longrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target} T \wedge (\llbracket S \rrbracket, T) \in \text{TRel})$

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

abbreviation (in *encoding*) *weakly-operational-sound* :: ('procT × 'procT) set ⇒ bool **where**
weakly-operational-sound TRel ≡
 $\forall S T. \llbracket S \rrbracket \mapsto \text{Target} * T \longrightarrow (\exists S' T'. S \mapsto \text{Source} * S' \wedge T \mapsto \text{Target} * T' \wedge (\llbracket S \rrbracket, T') \in \text{TRel})$

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

abbreviation (in *encoding*) *operational-sound* :: ('procT × 'procT) set ⇒ bool **where**
operational-sound TRel ≡ $\forall S T. \llbracket S \rrbracket \mapsto \text{Target} * T \longrightarrow (\exists S'. S \mapsto \text{Source} * S' \wedge (\llbracket S \rrbracket, T) \in \text{TRel})$

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

abbreviation (in *encoding*) *strongly-operational-sound* :: ('procT × 'procT) set ⇒ bool **where**
strongly-operational-sound TRel ≡
 $\forall S T. \llbracket S \rrbracket \mapsto \text{Target} T \longrightarrow (\exists S'. S \mapsto \text{Source} S' \wedge (\llbracket S \rrbracket, T) \in \text{TRel})$

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

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

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

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

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

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

8.1 Trivial Operational Correspondence Results

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

lemma (in *encoding*) *operational-correspondence-modulo-all-relation*:

shows *operational-complete* $\{(T1, T2). \text{True}\}$
and *weakly-operational-sound* $\{(T1, T2). \text{True}\}$
and *operational-sound* $\{(T1, T2). \text{True}\}$
 $\langle \text{proof} \rangle$

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

fixes *Cal* :: 'a *processCalculus*
shows *weak-reduction-bisimulation* $\{(a, b). \text{True}\}$ *Cal*
 $\langle \text{proof} \rangle$

lemma (in *encoding*) *operational-correspondence-modulo-some-target-relation*:

shows $\exists \text{TRel}. \text{weakly-operational-corresponding } \text{TRel}$
and $\exists \text{TRel}. \text{operational-corresponding } \text{TRel}$
and $\exists \text{TRel}. \text{weakly-operational-corresponding } \text{TRel} \wedge \text{weak-reduction-bisimulation } \text{TRel } \text{Target}$
and $\exists \text{TRel}. \text{operational-corresponding } \text{TRel} \wedge \text{weak-reduction-bisimulation } \text{TRel } \text{Target}$
 $\langle \text{proof} \rangle$

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 $(\exists \text{TRel}. \text{strongly-operational-corresponding } \text{TRel})$
 $= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$
and $(\exists \text{TRel}. \text{strongly-operational-corresponding } \text{TRel}$
 $\wedge \text{weak-reduction-bisimulation } \text{TRel } \text{Target})$
 $= (\forall S. (\exists S'. S \mapsto \text{Source } S') \longleftrightarrow (\exists T. \llbracket S \rrbracket \mapsto \text{Target } T))$

$\langle \text{proof} \rangle$

8.2 (Strong) Operational Completeness vs (Strong) Simulation

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

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

fixes *Rel* :: ('procS, 'procT) *Proc* \times ('procS, 'procT) *Proc* *set*
and *TRel* :: ('procT \times 'procT) *set*
assumes *A1*: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and *A2*: $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$
and *A3*: *weak-reduction-simulation* *Rel* (*STCal* *Source* *Target*)
shows *operational-complete* (*TRel*^{*})

$\langle \text{proof} \rangle$

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

fixes *TRel* :: ('procT \times 'procT) *set*
shows (*operational-complete* (*TRel*^{*})
 $\wedge \text{weak-reduction-simulation } (\text{TRel}^+) \text{Target}$)
 $= \text{weak-reduction-simulation } (\text{indRelRTPO } \text{TRel}) (\text{STCal } \text{Source } \text{Target})$

$\langle \text{proof} \rangle$

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

fixes *TRel* :: ('procT \times 'procT) *set*
shows (*operational-complete* (*TRel*^{*})
 $\wedge \text{weak-reduction-simulation } (\text{TRel}^+) \text{Target}$)
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$
 $\wedge \text{weak-reduction-simulation Rel (STCal Source Target)}$

$\langle \text{proof} \rangle$

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

lemma (in encoding) *strong-reduction-simulation-impl-SOCom*:

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

assumes A1: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

and A2: $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$

and A3: *strong-reduction-simulation Rel (STCal Source Target)*

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

$\langle \text{proof} \rangle$

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

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

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

\wedge *strong-reduction-simulation (TRel⁺) Target*

$=$ *strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)*

$\langle \text{proof} \rangle$

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

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

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

\wedge *strong-reduction-simulation (TRel⁺) Target*

$=$ ($\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$)

$\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$

\wedge *strong-reduction-simulation Rel (STCal Source Target)*

$\langle \text{proof} \rangle$

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

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

assumes stre: $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}$

$\longrightarrow (\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}^=$

shows $\exists \text{TRel}. (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$

$\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$

$\langle \text{proof} \rangle$

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

shows ($\exists \text{TRel}. \text{strongly-operational-complete (TRel*)}$)

\wedge *strong-reduction-simulation (TRel⁺) Target*

$=$ ($\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$)

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\text{TargetTerm } (\llbracket S \rrbracket), \text{TargetTerm } T) \in \text{Rel}^=)$

\wedge *strong-reduction-simulation Rel (STCal Source Target)*

$\langle \text{proof} \rangle$

8.3 Weak Operational Soundness vs Contrasimulation

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

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

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

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

assumes A1: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$

and $A2: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$
and $A3: \text{weak-reduction-contrasimulation } (\text{Rel}^{-1}) (\text{STCal Source Target})$
shows $\text{weakly-operational-sound } (\text{TRel}^*)$
 $\langle \text{proof} \rangle$

8.4 (Strong) Operational Soundness vs (Strong) Simulation

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

lemma (*in encoding*) $\text{weak-reduction-simulation-impl-OSou}$:
fixes $\text{Rel} :: ('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}$ set
and $\text{TRel} :: ('procT \times 'procT)$ set
assumes $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and $A2: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$
and $A3: \text{weak-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})$
shows $\text{operational-sound } (\text{TRel}^*)$
 $\langle \text{proof} \rangle$

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

lemma (*in encoding*) $\text{OSou-iff-weak-reduction-simulation}$:
fixes $\text{TRel} :: ('procT \times 'procT)$ set
shows $(\text{operational-sound } (\text{TRel}^*)$
 $\wedge \text{weak-reduction-simulation } ((\text{TRel}^+)^{-1}) \text{Target})$
 $= (\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel})$
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^+)$
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*)$
 $\wedge \text{weak-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target}))$
 $\langle \text{proof} \rangle$

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

lemma (*in encoding*) $\text{strong-reduction-simulation-impl-SOSou}$:
fixes $\text{Rel} :: ('procS, 'procT) \text{Proc} \times ('procS, 'procT) \text{Proc}$ set
and $\text{TRel} :: ('procT \times 'procT)$ set
assumes $A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and $A2: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel}^*$
and $A3: \text{strong-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})$
shows $\text{strongly-operational-sound } (\text{TRel}^*)$
 $\langle \text{proof} \rangle$

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

lemma (*in encoding*) $\text{SOSou-iff-strong-reduction-simulation}$:
fixes $\text{TRel} :: ('procT \times 'procT)$ set
shows $(\text{strongly-operational-sound } (\text{TRel}^*) \wedge \text{strong-reduction-simulation } ((\text{TRel}^+)^{-1}) \text{Target})$

$$\begin{aligned}
&= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \\
&\quad \wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \\
&\quad \wedge (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+) \\
&\quad \wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*) \\
&\quad \wedge \text{strong-reduction-simulation } (Rel^{-1}) \text{ (STCal Source Target)})
\end{aligned}$$

$\langle proof \rangle$

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

$$\begin{aligned}
&\text{shows } (\exists TRel. \text{strongly-operational-sound } (TRel^*) \\
&\quad \wedge \text{strong-reduction-simulation } ((TRel^+)^{-1}) \text{ Target}) \\
&= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \\
&\quad \wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm T) \in Rel^=) \\
&\quad \wedge \text{strong-reduction-simulation } (Rel^{-1}) \text{ (STCal Source Target)})
\end{aligned}$$

$\langle proof \rangle$

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*:

$$\begin{aligned}
&\text{fixes } Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc \text{ set} \\
&\quad \text{and } TRel :: ('procT \times 'procT) \text{ set} \\
&\text{assumes } enc: \forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel \\
&\quad \text{and } tRel: (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*) \\
&\quad \text{and } cs: \text{weak-reduction-correspondence-simulation } Rel \text{ (STCal Source Target)} \\
&\text{shows } \text{weakly-operational-corresponding } (TRel^*)
\end{aligned}$$

$\langle proof \rangle$

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

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

$$\begin{aligned}
&\text{fixes } TRel :: ('procT \times 'procT) \text{ set} \\
&\text{shows } (\text{weakly-operational-corresponding } (TRel^*) \\
&\quad \wedge \text{weak-reduction-correspondence-simulation } (TRel^+) \text{ Target}) \\
&= \text{weak-reduction-correspondence-simulation } (\text{indRelRTPO } TRel) \text{ (STCal Source Target)}
\end{aligned}$$

$\langle proof \rangle$

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

$$\begin{aligned}
&\text{fixes } TRel :: ('procT \times 'procT) \text{ set} \\
&\text{shows } (\text{weakly-operational-corresponding } (TRel^*) \\
&\quad \wedge \text{weak-reduction-correspondence-simulation } (TRel^+) \text{ Target}) \\
&= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \\
&\quad \wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \\
&\quad \wedge (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+) \\
&\quad \wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*) \\
&\quad \wedge \text{weak-reduction-correspondence-simulation } Rel \text{ (STCal Source Target)})
\end{aligned}$$

$\langle proof \rangle$

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

$$\begin{aligned}
&\text{fixes } Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc \text{ set} \\
&\quad \text{and } TRel :: ('procT \times 'procT) \text{ set} \\
&\text{assumes } \text{transT}: \text{trans } TRel \\
&\text{shows } ((\forall T1 T2. (T1, T2) \in TRel \longrightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \\
&\quad \wedge (\forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)) \\
&= (TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\})
\end{aligned}$$

$\langle proof \rangle$

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

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

shows (*weakly-operational-corresponding* $TRel \wedge \text{preorder } TRel$
 $\wedge \text{weak-reduction-correspondence-simulation } TRel \text{ Target}$
 $= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$
 $\wedge \text{preorder } Rel$
 $\wedge \text{weak-reduction-correspondence-simulation } Rel \text{ (STCal Source Target)})$

$\langle \text{proof} \rangle$

8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity

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

lemma (in *encoding*) *OC-iff-indRelRTPO-is-weak-reduction-bisimulation*:

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

shows (*operational-corresponding* $(TRel^*)$
 $\wedge \text{weak-reduction-bisimulation } (TRel^+) \text{ Target}$
 $= \text{weak-reduction-bisimulation } (\text{indRelRTPO } TRel) \text{ (STCal Source Target)}$

$\langle \text{proof} \rangle$

lemma (in *encoding*) *OC-iff-weak-reduction-bisimulation*:

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

shows (*operational-corresponding* $(TRel^*) \wedge \text{weak-reduction-bisimulation } (TRel^+) \text{ Target}$
 $= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$
 $\wedge (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)$
 $\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$
 $\wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)})$

$\langle \text{proof} \rangle$

lemma (in *encoding*) *OC-wrt-preorder-iff-weak-reduction-bisimulation*:

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

shows (*operational-corresponding* $TRel \wedge \text{preorder } TRel$
 $\wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$
 $= (\exists Rel. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$
 $\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$
 $\wedge \text{preorder } Rel$
 $\wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)})$

$\langle \text{proof} \rangle$

lemma (in *encoding*) *OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation*:

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

assumes $\text{eqT}: \text{equivalence } TRel$

shows (*operational-corresponding* $TRel \wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$) \longleftrightarrow
 $\text{weak-reduction-bisimulation } (\text{indRelTEQ } TRel) \text{ (STCal Source Target)}$

$\langle \text{proof} \rangle$

lemma (in *encoding*) *OC-wrt-equivalence-iff-weak-reduction-bisimulation*:

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

assumes $\text{eqT}: \text{equivalence } TRel$

shows (*operational-corresponding* $TRel \wedge \text{weak-reduction-bisimulation } TRel \text{ Target}$) \longleftrightarrow $(\exists Rel.$
 $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in Rel \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in Rel)$
 $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$
 $\wedge \text{trans } Rel \wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)})$

$\langle \text{proof} \rangle$

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

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
shows (*strongly-operational-corresponding* ($TRel^*$)
 \wedge *strong-reduction-bisimulation* ($TRel^+$) *Target*)
 $=$ *strong-reduction-bisimulation* ($indRelRTPO \ TRel$) (*STCal Source Target*)
 $\langle proof \rangle$

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
shows (*strongly-operational-corresponding* ($TRel^*$)
 \wedge *strong-reduction-bisimulation* ($TRel^+$) *Target*)
 $=$ ($\exists Rel. (\forall S. (SourceTerm \ S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$
 $\wedge (\forall T1 \ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm \ T1, TargetTerm \ T2) \in Rel)$
 $\wedge (\forall T1 \ T2. (TargetTerm \ T1, TargetTerm \ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$
 $\wedge (\forall S \ T. (SourceTerm \ S, TargetTerm \ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$
 \wedge *strong-reduction-bisimulation* Rel (*STCal Source Target*))
 $\langle proof \rangle$

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
shows (*strongly-operational-corresponding* $TRel \wedge$ *preorder* $TRel$
 \wedge *strong-reduction-bisimulation* $TRel \ Target$)
 $=$ ($\exists Rel. (\forall S. (SourceTerm \ S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
 $\wedge (\forall S \ T. (SourceTerm \ S, TargetTerm \ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$
 \wedge *preorder* Rel
 \wedge *strong-reduction-bisimulation* Rel (*STCal Source Target*))
 $\langle proof \rangle$

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

shows ($\exists TRel. \text{strongly-operational-corresponding} \ (TRel^*)$
 \wedge *strong-reduction-bisimulation* ($TRel^+$) *Target*)
 $=$ ($\exists Rel. (\forall S. (SourceTerm \ S, TargetTerm (\llbracket S \rrbracket)) \in Rel)$
 $\wedge (\forall S \ T. (SourceTerm \ S, TargetTerm \ T) \in Rel$
 $\longrightarrow (TargetTerm (\llbracket S \rrbracket), TargetTerm \ T) \in Rel^=)$
 \wedge *strong-reduction-bisimulation* Rel (*STCal Source Target*))
 $\langle proof \rangle$

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes eqT : *equivalence* $TRel$
shows (*strongly-operational-corresponding* $TRel \wedge$ *strong-reduction-bisimulation* $TRel \ Target$)
 \longleftrightarrow *strong-reduction-bisimulation* ($indRelTEQ \ TRel$) (*STCal Source Target*)
 $\langle proof \rangle$

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

fixes $TRel :: ('procT \times 'procT) \text{ set}$
assumes eqT : *equivalence* $TRel$
shows (*strongly-operational-corresponding* $TRel \wedge$ *strong-reduction-bisimulation* $TRel \ Target$)
 \longleftrightarrow ($\exists Rel.$
 $(\forall S. (SourceTerm \ S, TargetTerm (\llbracket S \rrbracket)) \in Rel \wedge (TargetTerm (\llbracket S \rrbracket), SourceTerm \ S) \in Rel)$
 $\wedge TRel = \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
 \wedge *trans* $Rel \wedge$ *strong-reduction-bisimulation* Rel (*STCal Source Target*))
 $\langle proof \rangle$

end

```

theory FullAbstraction
  imports SourceTargetRelation
begin

```

9 Full Abstraction

An encoding is fully abstract w.r.t. some source term relation SRel and some target term relation TRel if two source terms S1 and S2 form a pair (S1, S2) in SRel iff their literal translations form a pair (enc S1, enc S2) in TRel.

```

abbreviation (in encoding) fully-abstract
  :: ('procS × 'procS) set ⇒ ('procT × 'procT) set ⇒ bool
where
  fully-abstract SRel TRel ≡ ∀ S1 S2. (S1, S2) ∈ SRel ⟷ ([S1], [S2]) ∈ TRel

```

9.1 Trivial Full Abstraction Results

We start with some trivial full abstraction results. Each injective encoding is fully abstract w.r.t. to the identity relation on the source and the identity relation on the target.

```

lemma (in encoding) inj-enc-is-fully-abstract-wrt-identities:
  assumes injectivity: ∀ S1 S2. [S1] = [S2] ⟶ S1 = S2
  shows fully-abstract {(S1, S2). S1 = S2} {(T1, T2). T1 = T2}
  ⟨proof⟩

```

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

```

lemma (in encoding) fully-abstract-wrt-empty-relation:
  shows fully-abstract {} {}
  ⟨proof⟩

```

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

```

lemma (in encoding) fully-abstract-wrt-all-relation:
  shows fully-abstract {(S1, S2). True} {(T1, T2). True}
  ⟨proof⟩

```

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

```

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

```

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

```

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

```

```

lemma (in encoding) fully-abstract-wrt-trans-closure-of-source-relation:
  fixes RelS :: ('procS × 'procS) set
  assumes encRelS: ∀ S1 S2. [S1] = [S2] ⟶ (S1, S2) ∈ RelS+
  shows ∃ RelT. fully-abstract (RelS+) RelT
  ⟨proof⟩

```

For every encoding and every target term relation 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: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and srel: $\text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$
and trans: $\text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\})$
shows $\forall S1 S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel} \longleftrightarrow (\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}$

⟨*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: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and trel: $\forall S1 S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel} \longleftrightarrow (\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel}$
and trans: $\text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\})$
shows $\text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$

⟨*proof*⟩

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: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and trans: $\text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\})$
shows $(\forall S1 S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel} \longleftrightarrow (\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel})$
 $\longleftrightarrow (\text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\})$

⟨*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: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel}$
and encL: $\forall S. (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel}$
and trans: $\text{trans } \text{Rel}$
shows $(\forall S1 S2. (\llbracket S1 \rrbracket, \llbracket S2 \rrbracket) \in \text{TRel} \longleftrightarrow (\text{TargetTerm } (\llbracket S1 \rrbracket), \text{TargetTerm } (\llbracket S2 \rrbracket)) \in \text{Rel})$
 $\longleftrightarrow (\text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\})$

⟨*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 $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
shows $refl\ SRel \longleftrightarrow (\forall S. ([S], [S]) \in TRel)$
<proof>

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *surj*: $\forall T. \exists S. T = [S]$
shows $refl\ SRel \longleftrightarrow refl\ TRel$
<proof>

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
shows $sym\ SRel \longleftrightarrow sym\ \{(T1, T2). \exists S1\ S2. T1 = [S1] \wedge T2 = [S2] \wedge (T1, T2) \in TRel\}$
<proof>

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *surj*: $\forall T. \exists S. T = [S]$
shows $sym\ SRel \longleftrightarrow sym\ TRel$
<proof>

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
shows $trans\ SRel \longleftrightarrow trans\ \{(T1, T2). \exists S1\ S2. T1 = [S1] \wedge T2 = [S2] \wedge (T1, T2) \in TRel\}$
<proof>

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *surj*: $\forall T. \exists S. T = [S]$
shows $trans\ SRel \longleftrightarrow trans\ TRel$
<proof>

Similarly, a fully abstract encoding that respects a predicate ensures the this predicate is preserved, reflected, or respected by SRel iff it is preserved, reflected, or respected by TRel.

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-pred Pred*
shows $rel\text{-preserves-pred}\ \{(P, Q). \exists SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \wedge (SP, SQ) \in SRel\}\ Pred$
 $\longleftrightarrow rel\text{-preserves-pred}\ \{(P, Q). \exists SP\ SQ. [SP] \in T\ P \wedge [SQ] \in T\ Q \wedge ([SP], [SQ]) \in TRel\}\ Pred$
<proof>

lemma (in *encoding*) *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-preserve*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-binary-pred Pred*
shows $\text{rel-preserves-binary-pred } \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} \text{ Pred}$
 $\longleftrightarrow \text{rel-preserves-binary-pred}$
 $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \text{ Pred}$
<proof>

lemma (in *encoding*) *full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-pred Pred*
shows $\text{rel-reflects-pred } \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} \text{ Pred}$
 $\longleftrightarrow \text{rel-reflects-pred } \{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \text{ Pred}$
<proof>

lemma (in *encoding*) *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-binary-pred Pred*
shows $\text{rel-reflects-binary-pred } \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} \text{ Pred}$
 $\longleftrightarrow \text{rel-reflects-binary-pred}$
 $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \text{ Pred}$
<proof>

lemma (in *encoding*) *full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-respects*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-pred Pred*
shows $\text{rel-respects-pred } \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} \text{ Pred}$
 $\longleftrightarrow \text{rel-respects-pred } \{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \text{ Pred}$
<proof>

lemma (in *encoding*) *full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-respects*:
fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Pred :: ('procS, 'procT) \text{ Proc} \Rightarrow 'b \Rightarrow \text{bool}$
assumes *fullAbs*: *fully-abstract SRel TRel*
and *encP*: *enc-respects-binary-pred Pred*
shows $\text{rel-respects-binary-pred } \{(P, Q). \exists SP SQ. SP \in S P \wedge SQ \in S Q \wedge (SP, SQ) \in SRel\} \text{ Pred}$
 $\longleftrightarrow \text{rel-respects-binary-pred}$
 $\{(P, Q). \exists SP SQ. \llbracket SP \rrbracket \in T P \wedge \llbracket SQ \rrbracket \in T Q \wedge (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel\} \text{ Pred}$
<proof>

9.3 Full Abstraction w.r.t. Preorders

If there however exists a trans relation Rel that relates source terms and their literal translations in both directions, then the encoding is fully abstract with respect to the reduction of Rel to source terms and the reduction of Rel to target terms.

lemma (in *encoding*) *trans-source-target-relation-impl-full-abstraction*:

fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$ set
assumes $enc: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$
and $trans: trans\ Rel$
shows $fully\ abstract\ \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$
 $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
 $\langle proof \rangle$

lemma (*in encoding*) *source-target-relation-impl-full-abstraction-wrt-trans-closures:*
fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$ set
assumes $enc: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$
shows $fully\ abstract\ \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel^+\}$
 $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel^+\}$
 $\langle proof \rangle$

lemma (*in encoding*) *quasi-trans-source-target-relation-impl-full-abstraction:*
fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc)$ set
and $SRel :: ('procS \times 'procS)$ set
and $TRel :: ('procT \times 'procT)$ set
assumes $enc: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel$
and $srel: SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$
and $trel: TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
and $trans: \forall P\ Q\ R. (P, Q) \in Rel \wedge (Q, R) \in Rel \wedge ((P \in ProcS \wedge Q \in ProcT)$
 $\vee (P \in ProcT \wedge Q \in ProcS)) \longrightarrow (P, R) \in Rel$
shows $fully\ abstract\ SRel\ TRel$
 $\langle proof \rangle$

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

lemma (*in encoding*) *full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel:*
fixes $SRel :: ('procS \times 'procS)$ set
and $TRel :: ('procT \times 'procT)$ set
and $P\ Q :: ('procS, 'procT) Proc$
assumes $fullAbs: fully\ abstract\ SRel\ TRel$
and $rel: P \lesssim_{\llbracket \cdot \rrbracket} R < SRel, TRel > Q$
shows $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel^+$
and $\forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in TRel^*$
 $\langle proof \rangle$

lemma (*in encoding*) *full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel:*
fixes $SRel :: ('procS \times 'procS)$ set
and $TRel :: ('procT \times 'procT)$ set
and $P\ Q :: ('procS, 'procT) Proc$
assumes $fA: fully\ abstract\ SRel\ TRel$
and $transT: trans\ TRel$
and $reflS: refl\ SRel$
and $rel: P \sim_{\llbracket \cdot \rrbracket} R < SRel, TRel > Q$
shows $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (SP, SQ) \in SRel$
and $\forall SP\ SQ. SP \in S\ P \wedge SQ \in S\ Q \longrightarrow (\llbracket SP \rrbracket, \llbracket SQ \rrbracket) \in TRel$
and $\forall SP\ TQ. SP \in S\ P \wedge TQ \in T\ Q \longrightarrow (\llbracket SP \rrbracket, TQ) \in TRel$
and $\forall TP\ SQ. TP \in T\ P \wedge SQ \in S\ Q \longrightarrow (TP, \llbracket SQ \rrbracket) \in TRel$
and $\forall TP\ TQ. TP \in T\ P \wedge TQ \in T\ Q \longrightarrow (TP, TQ) \in TRel$
 $\langle proof \rangle$

If an encoding is fully abstract w.r.t. a preorder SRel on the source and a trans relation TRel on the target, then there exists a trans relation, namely indRelSTEQ, that relates source terms and their literal translations in both direction such that its reductions to source terms is SRel and its reduction to target terms is TRel.

lemma (in *encoding*) *full-abstraction-wrt-preorders-impl-trans-source-target-relation*:

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract* $SRel$ $TRel$
and $reflS$: *refl* $SRel$
and $transT$: *trans* $TRel$
shows $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$
 $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
 $\wedge trans\ Rel$

$\langle proof \rangle$

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*:

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
shows (*fully-abstract* $SRel$ $TRel \wedge refl\ SRel \wedge trans\ TRel) =$
 $(\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
 $\wedge (TargetTerm\ (\llbracket S \rrbracket), SourceTerm\ S) \in Rel)$
 $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in Rel\}$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
 $\wedge trans\ Rel)$

$\langle proof \rangle$

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*:

fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes *enc*: $\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel$
and $trans$: *trans* (*symcl* Rel)
shows *fully-abstract* $\{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel\}$
 $\{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in symcl\ Rel\}$

$\langle proof \rangle$

If an encoding is fully abstract w.r.t. the equivalences $SRel$ and $TRel$, then there exists a preorder, namely $indRelRSTPO$, that relates source terms and their literal translations such that its reductions to source terms is $SRel$ and its reduction to target terms is $TRel$.

lemma (in *encoding*) *fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder*:

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes *fullAbs*: *fully-abstract* $SRel$ $TRel$
and $reflT$: *refl* $TRel$
and $symmT$: *sym* $TRel$
and $transT$: *trans* $TRel$
shows $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge SRel = \{(S1, S2). (SourceTerm\ S1, SourceTerm\ S2) \in symcl\ Rel\}$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in symcl\ Rel\}$
 $\wedge preorder\ (symcl\ Rel)$

$\langle proof \rangle$

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

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$

assumes *fullAbs*: *fully-abstract* $((\text{symcl } (S\text{Rel}^=))^+)((\text{symcl } (T\text{Rel}^=))^+)$
shows $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge ((\text{symcl } (S\text{Rel}^=))^+ = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{symcl } \text{Rel}\})$
 $\wedge ((\text{symcl } (T\text{Rel}^=))^+ = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{symcl } \text{Rel}\})$
 $\wedge \text{preorder } (\text{symcl } \text{Rel})$

<proof>

lemma (*in encoding*) *fully-abstract-wrt-preorders-impl-source-target-relation-is-trans*:

fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
assumes *fullAbs*: *fully-abstract* *SRel* *TRel*
shows $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge S\text{Rel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$
 $\wedge T\text{Rel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$
 $\wedge ((\text{refl } S\text{Rel} \wedge \text{trans } T\text{Rel})$
 $\longleftrightarrow \text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\}))$

<proof>

lemma (*in encoding*) *fully-abstract-wrt-preorders-impl-source-target-relation-is-trans-B*:

fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
assumes *fullAbs*: *fully-abstract* *SRel* *TRel*
and *reflT*: *refl* *TRel*
and *transT*: *trans* *TRel*
shows $\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge S\text{Rel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$
 $\wedge T\text{Rel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$
 $\wedge \text{trans } (\text{Rel} \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T P \wedge S \in S Q\})$

<proof>

Thus an encoding is fully abstract w.r.t. an equivalence *SRel* on the source and an equivalence *TRel* on the target iff there exists a relation that relates source terms and their literal translations, whose sym closure is a preorder such that the reduction of this sym closure to source/target terms is *SRel*/*TRel*.

lemma (*in encoding*) *fully-abstract-wrt-equivalences-iff-symcl-source-target-relation-is-preorder*:

fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
shows (*fully-abstract* *SRel* *TRel* \wedge *equivalence* *TRel*) =
 $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge S\text{Rel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{symcl } \text{Rel}\}$
 $\wedge T\text{Rel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{symcl } \text{Rel}\}$
 $\wedge \text{preorder } (\text{symcl } \text{Rel}))$

<proof>

lemma (*in encoding*) *fully-abstract-iff-symcl-source-target-relation-is-preorder*:

fixes *SRel* :: ('procS × 'procS) set
and *TRel* :: ('procT × 'procT) set
shows *fully-abstract* $((\text{symcl } (S\text{Rel}^=))^+)((\text{symcl } (T\text{Rel}^=))^+) =$
 $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$
 $\wedge (\text{symcl } (S\text{Rel}^=))^+ = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{symcl } \text{Rel}\}$
 $\wedge (\text{symcl } (T\text{Rel}^=))^+ = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{symcl } \text{Rel}\}$
 $\wedge \text{preorder } (\text{symcl } \text{Rel}))$

<proof>

9.5 Full Abstraction without Relating Translations to their Source Terms

Let *Rel* be the result of removing from *indRelSTEQ* all pairs of two source or two target terms that are not contained in *SRel* or *TRel*. Then a fully abstract encoding ensures that *Rel* is *trans* iff *SRel* is *refl* and *TRel* is *trans*.

lemma (*in encoding*) *full-abstraction-impl-indRelSTEQ-is-trans*:

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
and $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes $fullAbs: \text{fully-abstract } SRel \ TRel$
and $rel: \quad Rel = ((indRelSTEQ \ SRel \ TRel)$
 $\quad - \{(P, Q). (P \in ProcS \wedge Q \in ProcS) \vee (P \in ProcT \wedge Q \in ProcT)\})$
 $\quad \cup \{(P, Q). (\exists SP \ SQ. SP \in S \ P \wedge SQ \in S \ Q \wedge (SP, SQ) \in SRel)$
 $\quad \quad \vee (\exists TP \ TQ. TP \in T \ P \wedge TQ \in T \ Q \wedge (TP, TQ) \in TRel)\}$
shows $(refl \ SRel \wedge trans \ TRel) = trans \ Rel$
 $\langle proof \rangle$

Whenever an encoding induces a trans relation that includes SRel and TRel and relates source terms to their literal translations in both directions, the encoding is fully abstract w.r.t. SRel and TRel.

lemma (in *encoding*) *trans-source-target-relation-impl-fully-abstract:*

fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
and $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes $enc: \quad \forall S. (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in Rel$
 $\quad \wedge (TargetTerm \ (\llbracket S \rrbracket), SourceTerm \ S) \in Rel$
and $srel: \quad SRel = \{(S1, S2). (SourceTerm \ S1, SourceTerm \ S2) \in Rel\}$
and $trel: \quad TRel = \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
and $trans: trans \ Rel$
shows *fully-abstract* $SRel \ TRel$
 $\langle proof \rangle$

Assume TRel is a preorder. Then an encoding is fully abstract w.r.t. SRel and TRel iff there exists a relation that relates add least all source terms to their literal translations, includes SRel and TRel, and whose union with the relation that relates exactly all literal translations to their source terms is trans.

lemma (in *encoding*) *source-target-relation-with-trans-impl-full-abstraction:*

fixes $Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \text{ set}$
assumes $enc: \quad \forall S. (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in Rel$
and $trans: trans \ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T \ P \wedge S \in S \ Q\})$
shows *fully-abstract* $\{(S1, S2). (SourceTerm \ S1, SourceTerm \ S2) \in Rel\}$
 $\quad \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
 $\langle proof \rangle$

lemma (in *encoding*) *fully-abstract-wrt-preorders-iff-source-target-relation-is-transB:*

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
assumes $preord: \text{preorder } TRel$
shows *fully-abstract* $SRel \ TRel =$
 $\quad (\exists Rel. (\forall S. (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in Rel)$
 $\quad \wedge SRel = \{(S1, S2). (SourceTerm \ S1, SourceTerm \ S2) \in Rel\}$
 $\quad \wedge TRel = \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
 $\quad \wedge trans \ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T \ P \wedge S \in S \ Q\}))$
 $\langle proof \rangle$

The same holds if to obtain transitivity the union may contain additional pairs that do neither relate two source nor two target terms.

lemma (in *encoding*) *fully-abstract-wrt-preorders-iff-source-target-relation-union-is-trans:*

fixes $SRel :: ('procS \times 'procS) \text{ set}$
and $TRel :: ('procT \times 'procT) \text{ set}$
shows $(\text{fully-abstract } SRel \ TRel \wedge refl \ SRel \wedge trans \ TRel) =$
 $\quad (\exists Rel. (\forall S. (SourceTerm \ S, TargetTerm \ (\llbracket S \rrbracket)) \in Rel)$
 $\quad \wedge SRel = \{(S1, S2). (SourceTerm \ S1, SourceTerm \ S2) \in Rel\}$
 $\quad \wedge TRel = \{(T1, T2). (TargetTerm \ T1, TargetTerm \ T2) \in Rel\}$
 $\quad \wedge (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \longleftrightarrow Q \in ProcT)$
 $\quad \quad \wedge trans \ (Rel \cup \{(P, Q). \exists S. \llbracket S \rrbracket \in T \ P \wedge S \in S \ Q\} \cup Rel'))$

<proof>

end

theory *CombinedCriteria*

imports *DivergenceReflection SuccessSensitiveness FullAbstraction OperationalCorrespondence*

begin

10 Combining Criteria

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

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

fixes *Cond* :: ('procS \Rightarrow 'procT) \Rightarrow bool

and *Pred* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool

assumes *Cond enc* = $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in Rel) \wedge Pred Rel)$

shows *Cond enc* = $(\exists Rel'. Pred (indRelR \cup Rel'))$

<proof>

lemma (*in encoding*) *combine-conditions-on-pairs-of-relations*:

fixes *RelA RelB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set

and *ConDA CondB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool

assumes $\forall (P, Q) \in RelA. ConDA (P, Q)$

and $\forall (P, Q) \in RelB. CondB (P, Q)$

shows $(\forall (P, Q) \in RelA \cap RelB. ConDA (P, Q)) \wedge (\forall (P, Q) \in RelA \cap RelB. CondB (P, Q))$

<proof>

lemma (*in encoding*) *combine-conditions-on-sets-of-relations*:

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

and *Cond* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool

and *ConDA* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool

assumes $\forall (P, Q) \in RelA. ConDA (P, Q)$

and *Cond Rel* \wedge *Rel* \subseteq *RelA*

shows *Cond Rel* \wedge $(\forall (P, Q) \in Rel. ConDA (P, Q))$

<proof>

lemma (*in encoding*) *combine-conditions-on-sets-and-pairs-of-relations*:

fixes *Rel RelA RelB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set

and *Cond* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool

and *ConDA CondB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool

assumes $\forall (P, Q) \in RelA. ConDA (P, Q)$

and $\forall (P, Q) \in RelB. CondB (P, Q)$

and *Cond Rel* \wedge *Rel* \subseteq *RelA* \wedge *Rel* \subseteq *RelB*

shows *Cond Rel* \wedge $(\forall (P, Q) \in Rel. ConDA (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB (P, Q))$

<proof>

We mapped several criteria on conditions on relations that relate at least all source terms and their literal translations. The following lemmata help us to combine such conditions by switching to the witness *indRelR*.

lemma (*in encoding*) *combine-conditions-on-relations-indRelR*:

fixes *RelA RelB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set

and *Cond* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool

and *ConDA CondB* :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool

assumes *A1*: $\forall S. (SourceTerm S, TargetTerm (\llbracket S \rrbracket)) \in RelA$

and *A2*: $\forall (P, Q) \in RelA. ConDA (P, Q)$

and $A3: \forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in RelB$
and $A4: \forall (P, Q) \in RelB. CondB\ (P, Q)$
shows $\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge (\forall (P, Q) \in Rel. CondA\ (P, Q))$
 $\wedge (\forall (P, Q) \in Rel. CondB\ (P, Q))$
and $Cond\ indRelR \implies (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge (\forall (P, Q) \in Rel. CondA\ (P, Q)) \wedge (\forall (P, Q) \in Rel. CondB\ (P, Q)) \wedge Cond\ Rel)$
 $\langle proof \rangle$

lemma (**in** *encoding*) *indRelR-cond-respects-predA-and-reflects-predB*:
fixes $PredA\ PredB :: ('procS, 'procT)\ Proc \Rightarrow bool$
shows $((\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA)$
 $\wedge (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-reflects-pred\ Rel\ PredB))$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel) \wedge rel-respects-pred\ Rel\ PredA$
 $\wedge rel-reflects-pred\ Rel\ PredB)$
 $\langle proof \rangle$

10.1 Divergence Reflection and Success Sensitiveness

We combine results on divergence reflection and success sensitiveness to analyse their combined effect on an encoding function. An encoding is success sensitive and reflects divergence iff there exists a relation that relates source terms and their literal translations that reflects divergence and respects success.

lemma (**in** *encoding-wrt-barbs*) *WSS-DR-iff-source-target-rel*:
fixes $success :: 'barbs$
shows $(enc-weakly-respects-barb-set\ \{success\} \wedge enc-reflects-divergence)$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel-weakly-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\})$
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target))$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *SS-DR-iff-source-target-rel*:
fixes $success :: 'barbs$
shows $(enc-respects-barb-set\ \{success\} \wedge enc-reflects-divergence)$
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge rel-respects-barb-set\ Rel\ (STCalWB\ SWB\ TWB)\ \{success\})$
 $\wedge rel-reflects-divergence\ Rel\ (STCal\ Source\ Target))$
 $\langle proof \rangle$

10.2 Adding Operational Correspondence

The effect of operational correspondence includes conditions (TRel is included, transitivity) that require a witness like indRelRTPO. In order to combine operational correspondence with success sensitiveness, we show that if the encoding and TRel (weakly) respects barbs than indRelRTPO (weakly) respects barbs. Since success is only a specific kind of barbs, the same holds for success sensitiveness.

lemma (**in** *encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-weakly-respects-success*:
fixes $success :: 'barbs$
and $TRel :: ('procT \times 'procT)\ set$
assumes $encRS: enc-weakly-respects-barb-set\ \{success\}$
and $trelPS: rel-weakly-preserves-barb-set\ TRel\ TWB\ \{success\}$
and $trelRS: rel-weakly-reflects-barb-set\ TRel\ TWB\ \{success\}$
shows $rel-weakly-respects-barb-set\ (indRelRTPO\ TRel)\ (STCalWB\ SWB\ TWB)\ \{success\}$
 $\langle proof \rangle$

lemma (**in** *encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs*:
fixes $TRel :: ('procT \times 'procT)\ set$
assumes $encRS: enc-weakly-respects-barbs$
and $trelPS: rel-weakly-preserves-barbs\ TRel\ TWB$
and $trelRS: rel-weakly-reflects-barbs\ TRel\ TWB$
shows $rel-weakly-respects-barbs\ (indRelRTPO\ TRel)\ (STCalWB\ SWB\ TWB)$

<proof>

lemma (in *encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-respects-success*:
fixes *success* :: 'barbs
and *TRel* :: ('procT × 'procT) set
assumes *encRS*: *enc-respects-barb-set* {*success*}
and *treIPS*: *rel-preserves-barb-set* *TRel* *TWB* {*success*}
and *treIRS*: *rel-reflects-barb-set* *TRel* *TWB* {*success*}
shows *rel-respects-barb-set* (*indRelRTPO* *TRel*) (*STCalWB* *SWB* *TWB*) {*success*}
<proof>

lemma (in *encoding-wrt-barbs*) *enc-and-TRel-impl-indRelRTPO-respects-barbs*:
fixes *TRel* :: ('procT × 'procT) set
assumes *encRS*: *enc-respects-barbs*
and *treIPS*: *rel-preserves-barbs* *TRel* *TWB*
and *treIRS*: *rel-reflects-barbs* *TRel* *TWB*
shows *rel-respects-barbs* (*indRelRTPO* *TRel*) (*STCalWB* *SWB* *TWB*)
<proof>

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

lemma (in *encoding-wrt-barbs*) *OC-SS-iff-source-target-rel*:
fixes *success* :: 'barbs
and *TRel* :: ('procT × 'procT) set
shows (*operational-corresponding* (*TRel*^{*})
∧ *weak-reduction-bisimulation* (*TRel*⁺) *Target*
∧ *enc-weakly-respects-barb-set* {*success*}
∧ *rel-weakly-respects-barb-set* *TRel* *TWB* {*success*})
= (∃ *Rel*. (∀ *S*. (*SourceTerm* *S*, *TargetTerm* ($\llbracket S \rrbracket$)) ∈ *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* → ($\llbracket S \rrbracket$, *T*) ∈ *TRel*^{*})
∧ *weak-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)
∧ *rel-weakly-respects-barb-set* *Rel* (*STCalWB* *SWB* *TWB*) {*success*})
<proof>

lemma (in *encoding-wrt-barbs*) *OC-SS-RB-iff-source-target-rel*:
fixes *success* :: 'barbs
and *TRel* :: ('procT × 'procT) set
shows (*operational-corresponding* (*TRel*^{*})
∧ *weak-reduction-bisimulation* (*TRel*⁺) *Target*
∧ *enc-weakly-respects-barbs* ∧ *enc-weakly-respects-barb-set* {*success*}
∧ *rel-weakly-respects-barbs* *TRel* *TWB* ∧ *rel-weakly-respects-barb-set* *TRel* *TWB* {*success*})
= (∃ *Rel*. (∀ *S*. (*SourceTerm* *S*, *TargetTerm* ($\llbracket S \rrbracket$)) ∈ *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* → ($\llbracket S \rrbracket$, *T*) ∈ *TRel*^{*})
∧ *weak-reduction-bisimulation* *Rel* (*STCal* *Source* *Target*)
∧ *rel-weakly-respects-barbs* *Rel* (*STCalWB* *SWB* *TWB*)
∧ *rel-weakly-respects-barb-set* *Rel* (*STCalWB* *SWB* *TWB*) {*success*})
<proof>

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

$$\begin{aligned}
& \wedge TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} \\
& \wedge (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel) \\
& \wedge \text{weak-reduction-bisimulation } Rel \text{ (STCal Source Target)} \wedge \text{preorder } Rel \\
& \wedge \text{rel-weakly-respects-barb-set } Rel \text{ (STCalWB SWB TWB) \{success\}}
\end{aligned}$$

<proof>

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

fixes *success* :: 'barbs

and *TRel* :: ('procT × 'procT) set

shows (*operational-corresponding TRel* ∧ *preorder TRel* ∧ *weak-reduction-bisimulation TRel Target* ∧ *enc-weakly-respects-barbs* ∧ *rel-weakly-respects-barbs TRel TWB*

∧ *enc-weakly-respects-barb-set* {*success*}

∧ *rel-weakly-respects-barb-set TRel TWB* {*success*})

= (∃ *Rel*. (∀ *S*. (SourceTerm *S*, TargetTerm (⌊*S*⌋)) ∈ *Rel*)

∧ *TRel* = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ *Rel*}

∧ (∀ *S T*. (SourceTerm *S*, TargetTerm *T*) ∈ *Rel* → (⌊*S*⌋, *T*) ∈ *TRel*)

∧ *weak-reduction-bisimulation Rel* (STCal Source Target) ∧ *preorder Rel*

∧ *rel-weakly-respects-barbs Rel* (STCalWB SWB TWB)

∧ *rel-weakly-respects-barb-set Rel* (STCalWB SWB TWB) {*success*})

<proof>

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

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

fixes *success* :: 'barbs

and *TRel* :: ('procT × 'procT) set

shows (*weakly-operational-corresponding TRel* ∧ *preorder TRel*

∧ *weak-reduction-correspondence-simulation TRel Target*

∧ *enc-weakly-respects-barb-set* {*success*}

∧ *rel-weakly-respects-barb-set TRel TWB* {*success*})

= (∃ *Rel*. (∀ *S*. (SourceTerm *S*, TargetTerm (⌊*S*⌋)) ∈ *Rel*)

∧ *TRel* = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ *Rel*}

∧ (∀ *S T*. (SourceTerm *S*, TargetTerm *T*) ∈ *Rel* → (⌊*S*⌋, *T*) ∈ *TRel*)

∧ *weak-reduction-correspondence-simulation Rel* (STCal Source Target) ∧ *preorder Rel*

∧ *rel-weakly-respects-barb-set Rel* (STCalWB SWB TWB) {*success*})

<proof>

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

fixes *success* :: 'barbs

and *TRel* :: ('procT × 'procT) set

shows (*weakly-operational-corresponding TRel* ∧ *preorder TRel*

∧ *weak-reduction-correspondence-simulation TRel Target*

∧ *enc-weakly-respects-barbs* ∧ *enc-weakly-respects-barb-set* {*success*}

∧ *rel-weakly-respects-barbs TRel TWB* ∧ *rel-weakly-respects-barb-set TRel TWB* {*success*})

= (∃ *Rel*. (∀ *S*. (SourceTerm *S*, TargetTerm (⌊*S*⌋)) ∈ *Rel*)

∧ *TRel* = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ *Rel*}

∧ (∀ *S T*. (SourceTerm *S*, TargetTerm *T*) ∈ *Rel* → (⌊*S*⌋, *T*) ∈ *TRel*)

∧ *weak-reduction-correspondence-simulation Rel* (STCal Source Target) ∧ *preorder Rel*

∧ *rel-weakly-respects-barbs Rel* (STCalWB SWB TWB)

∧ *rel-weakly-respects-barb-set Rel* (STCalWB SWB TWB) {*success*})

<proof>

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

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

fixes *success* :: 'barbs

and $TRel \quad :: ('procT \times 'procT) \text{ set}$
shows (*strongly-operational-corresponding* $TRel \wedge preorder\ TRel$
 \wedge *strong-reduction-bisimulation* $TRel\ Target$
 \wedge *enc-respects-barb-set* $\{success\} \wedge$ *rel-respects-barb-set* $TRel\ TWB\ \{success\}$)
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$
 \wedge *strong-reduction-bisimulation* $Rel\ (STCal\ Source\ Target) \wedge$ *preorder* Rel
 \wedge *rel-respects-barb-set* $Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$)
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *SOC-SS-RB-wrt-preorder-iff-source-target-rel*:
fixes $success \quad :: 'barbs$
and $TRel \quad :: ('procT \times 'procT) \text{ set}$
shows (*strongly-operational-corresponding* $TRel \wedge preorder\ TRel$
 \wedge *strong-reduction-bisimulation* $TRel\ Target$
 \wedge *enc-respects-barbs* \wedge *rel-respects-barbs* $TRel\ TWB$
 \wedge *enc-respects-barb-set* $\{success\} \wedge$ *rel-respects-barb-set* $TRel\ TWB\ \{success\}$)
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel)$
 \wedge *strong-reduction-bisimulation* $Rel\ (STCal\ Source\ Target) \wedge$ *preorder* Rel
 \wedge *rel-respects-barbs* $Rel\ (STCalWB\ SWB\ TWB)$
 \wedge *rel-respects-barb-set* $Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$)
 $\langle proof \rangle$

Next we also add divergence reflection to operational correspondence and success sensitiveness.

lemma (*in encoding*) *enc-and-TRelimpl-indRelRTPO-reflect-divergence*:
fixes $TRel \quad :: ('procT \times 'procT) \text{ set}$
assumes *encRD*: *enc-reflects-divergence*
and *treIRD*: *rel-reflects-divergence* $TRel\ Target$
shows *rel-reflects-divergence* (*indRelRTPO* $TRel$) (*STCal* $Source\ Target$)
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *OC-SS-DR-iff-source-target-rel*:
fixes $success \quad :: 'barbs$
and $TRel \quad :: ('procT \times 'procT) \text{ set}$
shows (*operational-corresponding* $(TRel^*)$
 \wedge *weak-reduction-bisimulation* $(TRel^+)\ Target$
 \wedge *enc-weakly-respects-barb-set* $\{success\}$
 \wedge *rel-weakly-respects-barb-set* $TRel\ TWB\ \{success\}$
 \wedge *enc-reflects-divergence* \wedge *rel-reflects-divergence* $TRel\ Target$)
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge (\forall T1\ T2. (T1, T2) \in TRel \longrightarrow (TargetTerm\ T1, TargetTerm\ T2) \in Rel)$
 $\wedge (\forall T1\ T2. (TargetTerm\ T1, TargetTerm\ T2) \in Rel \longrightarrow (T1, T2) \in TRel^+)$
 $\wedge (\forall S\ T. (SourceTerm\ S, TargetTerm\ T) \in Rel \longrightarrow (\llbracket S \rrbracket, T) \in TRel^*)$
 \wedge *weak-reduction-bisimulation* $Rel\ (STCal\ Source\ Target)$
 \wedge *rel-weakly-respects-barb-set* $Rel\ (STCalWB\ SWB\ TWB)\ \{success\}$
 \wedge *rel-reflects-divergence* $Rel\ (STCal\ Source\ Target)$)
 $\langle proof \rangle$

lemma (*in encoding-wrt-barbs*) *WOC-SS-DR-wrt-preorder-iff-source-target-rel*:
fixes $success \quad :: 'barbs$
and $TRel \quad :: ('procT \times 'procT) \text{ set}$
shows (*weakly-operational-corresponding* $TRel \wedge preorder\ TRel$
 \wedge *weak-reduction-correspondence-simulation* $TRel\ Target$
 \wedge *enc-weakly-respects-barb-set* $\{success\}$
 \wedge *rel-weakly-respects-barb-set* $TRel\ TWB\ \{success\}$
 \wedge *enc-reflects-divergence* \wedge *rel-reflects-divergence* $TRel\ Target$)
 $= (\exists Rel. (\forall S. (SourceTerm\ S, TargetTerm\ (\llbracket S \rrbracket)) \in Rel)$
 $\wedge TRel = \{(T1, T2). (TargetTerm\ T1, TargetTerm\ T2) \in Rel\}$)

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$
 $\wedge \text{weak-reduction-correspondence-simulation } \text{Rel } (\text{STCal } \text{Source } \text{Target}) \wedge \text{preorder } \text{Rel}$
 $\wedge \text{rel-weakly-respects-barb-set } \text{Rel } (\text{STCalWB } \text{SWB } \text{TWB}) \{ \text{success} \}$
 $\wedge \text{rel-reflects-divergence } \text{Rel } (\text{STCal } \text{Source } \text{Target})$

$\langle \text{proof} \rangle$

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

fixes *success* :: 'barbs

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

shows (*operational-corresponding* *TRel* \wedge *preorder* *TRel* \wedge *weak-reduction-bisimulation* *TRel* *Target*

\wedge *enc-weakly-respects-barb-set* {*success*}

\wedge *rel-weakly-respects-barb-set* *TRel* *TWB* {*success*}

\wedge *enc-reflects-divergence* \wedge *rel-reflects-divergence* *TRel* *Target*)

= $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$

$\wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target}) \wedge \text{preorder } \text{Rel}$

$\wedge \text{rel-weakly-respects-barb-set } \text{Rel } (\text{STCalWB } \text{SWB } \text{TWB}) \{ \text{success} \}$

$\wedge \text{rel-reflects-divergence } \text{Rel } (\text{STCal } \text{Source } \text{Target})$)

$\langle \text{proof} \rangle$

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

fixes *success* :: 'barbs

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

shows (*strongly-operational-corresponding* *TRel* \wedge *preorder* *TRel*

\wedge *strong-reduction-bisimulation* *TRel* *Target*

\wedge *enc-respects-barb-set* {*success*} \wedge *rel-respects-barb-set* *TRel* *TWB* {*success*}

\wedge *enc-reflects-divergence* \wedge *rel-reflects-divergence* *TRel* *Target*)

= $(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel})$

$\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

$\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow (\llbracket S \rrbracket, T) \in \text{TRel})$

$\wedge \text{strong-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target}) \wedge \text{preorder } \text{Rel}$

$\wedge \text{rel-respects-barb-set } \text{Rel } (\text{STCalWB } \text{SWB } \text{TWB}) \{ \text{success} \}$

$\wedge \text{rel-reflects-divergence } \text{Rel } (\text{STCal } \text{Source } \text{Target})$)

$\langle \text{proof} \rangle$

10.3 Full Abstraction and Operational Correspondence

To combine full abstraction and operational correspondence we consider a symmetric version of the induced relation and assume that the relations *SRel* and *TRel* are equivalences. Then an encoding is fully abstract w.r.t. *SRel* and *TRel* and operationally corresponding w.r.t. *TRel* such that *TRel* is a bisimulation iff the induced relation contains both *SRel* and *TRel* and is a transitive bisimulation.

lemma (in *encoding*) *FS-OC-modulo-equivalences-iff-source-target-relation*:

fixes *SRel* :: ('procS \times 'procS) set

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

assumes *eqS*: *equivalence* *SRel*

and *eqT*: *equivalence* *TRel*

shows *fully-abstract* *SRel* *TRel*

\wedge *operational-corresponding* *TRel* \wedge *weak-reduction-bisimulation* *TRel* *Target*

$\longleftrightarrow (\exists \text{Rel}.$

$(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel} \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel})$

$\wedge \text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$

$\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$

$\wedge \text{trans } \text{Rel} \wedge \text{weak-reduction-bisimulation } \text{Rel } (\text{STCal } \text{Source } \text{Target})$)

$\langle \text{proof} \rangle$

lemma (in *encoding*) *FA-SOC-modulo-equivalences-iff-source-target-relation*:

fixes *SRel* :: ('procS \times 'procS) set

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

assumes *eqS*: *equivalence* *SRel*

and *eqT*: equivalence *TRel*
shows *fully-abstract SRel TRel* \wedge *strongly-operational-corresponding TRel*
 \wedge *strong-reduction-bisimulation TRel Target* \longleftrightarrow $(\exists \text{Rel}.$
 $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } (\llbracket S \rrbracket)) \in \text{Rel} \wedge (\text{TargetTerm } (\llbracket S \rrbracket), \text{SourceTerm } S) \in \text{Rel})$
 $\wedge \text{SRel} = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}$
 $\wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\} \wedge \text{trans Rel}$
 $\wedge \text{strong-reduction-bisimulation Rel } (\text{STCal Source Target}))$
 $\langle \text{proof} \rangle$

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

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

fixes *SRel* :: ('procS \times 'procS) set
and *TRel* :: ('procT \times 'procT) set
assumes *fullAbs*: *fully-abstract SRel TRel*
and *opCom*: *operational-complete TRel*
and *opSou*: *operational-sound TRel*
and *symmT*: *sym TRel*
and *transT*: *trans TRel*
and *bisimT*: *weak-reduction-bisimulation TRel Target*
shows *weak-reduction-bisimulation SRel Source*
 $\langle \text{proof} \rangle$

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

fixes *SRel* :: ('procS \times 'procS) set
and *TRel* :: ('procT \times 'procT) set
assumes *fullAbs*: *fully-abstract SRel TRel*
and *opCom*: *strongly-operational-complete TRel*
and *opSou*: *strongly-operational-sound TRel*
and *symmT*: *sym TRel*
and *transT*: *trans TRel*
and *bisimT*: *strong-reduction-bisimulation TRel Target*
shows *strong-reduction-bisimulation SRel Source*
 $\langle \text{proof} \rangle$

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

fixes *SRel* :: ('procS \times 'procS) set
and *TRel* :: ('procT \times 'procT) set
assumes *fullAbs*: *fully-abstract SRel TRel*
and *opCor*: *operational-corresponding TRel*
and *symmT*: *sym TRel*
and *transT*: *trans TRel*
and *surj*: $\forall T. \exists S. T = \llbracket S \rrbracket$
shows *weak-reduction-bisimulation SRel Source* \longleftrightarrow *weak-reduction-bisimulation TRel Target*
 $\langle \text{proof} \rangle$

end