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

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Relations</td>
<td></td>
</tr>
<tr>
<td></td>
<td>1.1 Basic Conditions</td>
<td>3</td>
</tr>
<tr>
<td></td>
<td>1.2 Preservation, Reflection, and Respection of Predicates</td>
<td>6</td>
</tr>
<tr>
<td>2</td>
<td>Process Calculi</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2.1 Reduction Semantics</td>
<td>11</td>
</tr>
<tr>
<td></td>
<td>2.1.1 Observables or Barbs</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>Simulation Relations</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>3.1 Simulation</td>
<td>17</td>
</tr>
<tr>
<td></td>
<td>3.2 Contrasimulation</td>
<td>22</td>
</tr>
<tr>
<td></td>
<td>3.3 Coupled Simulation</td>
<td>24</td>
</tr>
<tr>
<td></td>
<td>3.4 Correspondence Simulation</td>
<td>25</td>
</tr>
<tr>
<td></td>
<td>3.5 Bisimulation</td>
<td>30</td>
</tr>
<tr>
<td></td>
<td>3.6 Step Closure of Relations</td>
<td>40</td>
</tr>
<tr>
<td>4</td>
<td>Encodings</td>
<td>46</td>
</tr>
<tr>
<td>5</td>
<td>Relation between Source and Target Terms</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>5.1 Relations Induced by the Encoding Function</td>
<td>54</td>
</tr>
<tr>
<td></td>
<td>5.2 Relations Induced by the Encoding and a Relation on Target Terms</td>
<td>77</td>
</tr>
<tr>
<td></td>
<td>5.3 Relations Induced by the Encoding and Relations on Source Terms and Target Terms</td>
<td>120</td>
</tr>
<tr>
<td>6</td>
<td>Success Sensitiveness and Barbs</td>
<td>151</td>
</tr>
<tr>
<td>7</td>
<td>Divergence Reflection</td>
<td>155</td>
</tr>
<tr>
<td>8</td>
<td>Operational Correspondence</td>
<td>157</td>
</tr>
<tr>
<td></td>
<td>8.1 Trivial Operational Correspondence Results</td>
<td>158</td>
</tr>
<tr>
<td></td>
<td>8.2 (Strong) Operational Completeness vs (Strong) Simulation</td>
<td>159</td>
</tr>
<tr>
<td></td>
<td>8.3 Weak Operational Soundness vs Contrasimulation</td>
<td>167</td>
</tr>
<tr>
<td></td>
<td>8.4 (Strong) Operational Soundness vs (Strong) Simulation</td>
<td>168</td>
</tr>
<tr>
<td></td>
<td>8.5 Weak Operational Correspondence vs Correspondence Similarity</td>
<td>175</td>
</tr>
<tr>
<td></td>
<td>8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity</td>
<td>181</td>
</tr>
<tr>
<td>9</td>
<td>Full Abstraction</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>9.1 Trivial Full Abstraction Results</td>
<td>201</td>
</tr>
<tr>
<td></td>
<td>9.2 Fully Abstract Encodings</td>
<td>203</td>
</tr>
<tr>
<td></td>
<td>9.3 Full Abstraction w.r.t. Preorders</td>
<td>210</td>
</tr>
<tr>
<td></td>
<td>9.4 Full Abstraction w.r.t. Equivalences</td>
<td>218</td>
</tr>
<tr>
<td></td>
<td>9.5 Full Abstraction without Relating Translations to their Source Terms</td>
<td>228</td>
</tr>
<tr>
<td>10</td>
<td>Combining Criteria</td>
<td>235</td>
</tr>
<tr>
<td></td>
<td>10.1 Divergence Reflection and Success Sensitiveness</td>
<td>237</td>
</tr>
<tr>
<td></td>
<td>10.2 Adding Operational Correspondence</td>
<td>238</td>
</tr>
<tr>
<td></td>
<td>10.3 Full Abstraction and Operational Correspondence</td>
<td>272</td>
</tr>
</tbody>
</table>
theory Relations
imports Main Order-Relation ~/src/HOL/Library/LaTeXsugar ~/src/HOL/Library/OptionalSugar
begin

1 Relations
1.1 Basic Conditions
We recall the standard definitions for reflexivity, symmetry, transitivity, preorders, equivalence, and inverse relations.

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

A symmetric preorder is an equivalence.
lemma symm-preorder-is-equivalence:
  fixes Rel :: ('a × 'a) set
  assumes preorder Rel and sym Rel
  shows equivalence Rel
  using assms unfolding preorder-on-def equiv-def
  by simp

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

definition symcl :: ('a × 'a) set ⇒ ('a × 'a) set where
  symcl Rel = Rel ∪ Rel⁻¹

For all (a, b) in R, the symmetric closure of R contains (a, b) as well as (b, a).
lemma elem-of-symcl:
  fixes Rel :: ('a × 'a) set and a b :: 'a
  assumes elem: (a, b) ∈ Rel
  shows (a, b) ∈ symcl Rel and (b, a) ∈ symcl Rel
  by (auto simp add: elem symcl-def)

The symmetric closure of a relation is symmetric.
lemma sym-symcl:
  fixes Rel :: ('a × 'a) set
  shows sym (symcl Rel)
  by (simp add: symcl-def sym-Un-converse)

The reflexive and symmetric closure of a relation is equal to its symmetric and reflexive closure.
lemma refl-symm-closure-is-symm-refl-closure:
  fixes Rel :: ('a × 'a) set
  shows symcl (Rel) = (symcl Rel)
  by (auto simp add: symcl-def refl)

The reflexive 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)
  using assms
  by (auto simp add: refl-on-def symcl-def)
Accordingly, the reflexive, symmetric, and transitive closure of a relation is equal to its symmetric, reflexive, and transitive closure.

**lemma refl-symm-trans-closure-is-symm-refl-trans-closure:**
fixes Rel :: ('a × 'a) set
shows (symcl (Rel^=))^+ = (symcl Rel)^+
using refl-symm-closure-is-symm-refl-closure[where Rel=Rel]
by simp

The reflexive closure of a symmetric relation is symmetric.

**lemma sym-reflcl-of-symm-rel:**
fixes Rel :: ('a × 'a) set
assumes sym Rel
shows sym (Rel^=)
using assms
by (simp add: sym-Id sym-Un)

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

**lemma reflcl-of-refl-rel:**
fixes Rel :: ('a × 'a) set
assumes refl Rel
shows Rel = Rel
using assms unfolding refl-on-def
by auto

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

**lemma symm-closure-of-symm-rel:**
fixes Rel :: ('a × 'a) set
assumes sym Rel
shows symcl Rel = Rel
using assms unfolding symcl-def sym-def
by auto

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

**lemma rtrancl-of-preorder:**
fixes Rel :: ('a × 'a) set
assumes preorder Rel
shows Rel^* = Rel
using assms reflcl-of-refl-rel[of Rel] trancl-id[of Rel^=] trancl-reflcl[of Rel]
unfolding preorder-on-def
by auto

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

**lemma refl-trans-closure-subset-of-refl-symm-trans-closure:**
fixes Rel :: ('a × 'a) set
shows Rel^* ⊆ (symcl (Rel^=))^+
proof clarify
fix a b
assume (a, b) ∈ Rel^*
hence (a, b) ∈ (symcl Rel)^*
using in-rtrancl-UnI[of (a, b) Rel Rel^-1]
by (simp add: symcl-def)
thus (a, b) ∈ (symcl (Rel^=))^+
using refl-symm-trans-closure-is-symm-refl-trans-closure[of Rel]
by simp
qed
If a preorder Rel satisfies the following two conditions, then its symmetric closure is transitive: (1) If
(a, b) and (c, b) in Rel but not (a, c) in Rel, then (b, a) in Rel or (b, c) in Rel. (2) If (a, b) and (a, c)
in Rel but not (b, c) in Rel, then (b, a) in Rel or (c, a) in Rel.

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

- **Fixes** Rel :: ('a × 'a) set
- **Assumes** condA: ∀ a b c. (a, b) ∈ Rel ∧ (c, b) ∈ Rel ∧ (a, c) ∉ Rel
  - ▶️ (b, a) ∈ Rel ∨ (b, c) ∈ Rel
- and condB: ∀ a b c. (a, b) ∈ Rel ∧ (a, c) ∈ Rel ∧ (b, c) ∉ Rel
  - ▶️ (b, a) ∈ Rel ∨ (c, a) ∈ Rel
- and reflR: refl Rel
- and tranR: trans Rel
- **Shows** trans (symcl Rel)

**Unfolding** trans-def

**Proof** clarify

- **Fix** a b c
- **Have** [(a, b) ∈ Rel; (b, c) ∈ Rel] ▶️ (a, c) ∈ symcl Rel
- **Proof**
  - **Assume** (a, b) ∈ Rel and (b, c) ∈ Rel
  - **With** tranR **Have** (a, c) ∈ Rel
    - **Unfolding** trans-def
      - **By** blast
      - **Thus** (a, c) ∈ symcl Rel
        - **By** (simp add: symcl-def)
- **Qed**

- **Moreover** have [(a, b) ∈ Rel; (c, b) ∈ Rel; (a, c) ∉ Rel] ▶️ (a, c) ∈ symcl Rel
- **Proof**
  - **Assume** A1: (a, b) ∈ Rel and A2: (c, b) ∈ Rel and (a, c) ∉ Rel
  - **With** condA **Have** (b, a) ∈ Rel ∨ (b, c) ∈ Rel
    - **By** blast
    - **Thus** (a, c) ∈ symcl Rel
      - **Proof** auto
        - **Assume** (b, a) ∈ Rel
          - **With** A2 tranR **Have** (c, a) ∈ Rel
            - **Unfolding** trans-def
              - **By** blast
              - **Thus** (a, c) ∈ symcl Rel
                - **By** (simp add: symcl-def)
- **Next**
  - **Assume** (b, c) ∈ Rel
  - **With** A1 tranR **Have** (a, c) ∈ Rel
    - **Unfolding** trans-def
      - **By** blast
      - **Thus** (a, c) ∈ symcl Rel
        - **By** (simp add: symcl-def)
- **Qed**

- **Moreover** have [(b, a) ∈ Rel; (b, c) ∈ Rel; (a, c) ∉ Rel] ▶️ (a, c) ∈ symcl Rel
- **Proof**
  - **Assume** B1: (b, a) ∈ Rel and B2: (b, c) ∈ Rel and (a, c) ∉ Rel
  - **With** condB **Have** (a, b) ∈ Rel ∨ (c, b) ∈ Rel
    - **By** blast
    - **Thus** (a, c) ∈ symcl Rel
      - **Proof** auto
        - **Assume** (a, b) ∈ Rel
          - **With** B2 tranR **Have** (a, c) ∈ Rel
            - **Unfolding** trans-def
              - **By** blast
              - **Thus** (a, c) ∈ symcl Rel
                - **By** (simp add: symcl-def)
- **Next**
assume \((c, b) \in \text{Rel}\)
with \(B1 \text{ tranR have } (c, a) \in \text{Rel}\)
  unfolding trans-def
  by blast
  thus \((a, c) \in \text{symcl Rel}\)
  by (simp add: symcl-def)
qed

moreover have \([[(b, a) \in \text{Rel}; (c, b) \in \text{Rel}] \implies (a, c) \in \text{symcl Rel}]\)
proof –
  assume \((c, b) \in \text{Rel} \land (b, a) \in \text{Rel}\)
  with tranR have \((c, a) \in \text{Rel}\)
    unfolding trans-def
    by blast
    thus \((a, c) \in \text{symcl Rel}\)
    by (simp add: symcl-def)
qed

moreover assume \((a, b) \in \text{symcl Rel} \land (b, c) \in \text{symcl Rel}\)
ultimately show \((a, c) \in \text{symcl Rel}\)
  by (auto simp add: symcl-def)
qed

1.2 Preservation, Reflection, and Respection of Predicates

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

abbreviation rel-preserves-pred :: \((\alpha \times \beta)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-preserves-pred \(\text{Rel Pred} \equiv \forall a b. (a, b) \in \text{Rel} \land \text{Pred a} \longrightarrow \text{Pred b}\)

abbreviation rel-preserves-binary-pred :: \((\alpha \times \alpha)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-preserves-binary-pred \(\text{Rel Pred} \equiv \forall a b x. (a, b) \in \text{Rel} \land \text{Pred a x} \longrightarrow \text{Pred b x}\)

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

abbreviation rel-reflects-pred :: \((\alpha \times \beta)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-reflects-pred \(\text{Rel Pred} \equiv \forall a b. (a, b) \in \text{Rel} \land \text{Pred b} \longrightarrow \text{Pred a}\)

abbreviation rel-reflects-binary-pred :: \((\alpha \times \alpha)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-reflects-binary-pred \(\text{Rel Pred} \equiv \forall a b x. (a, b) \in \text{Rel} \land \text{Pred b x} \longrightarrow \text{Pred a x}\)

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

abbreviation rel-respects-pred :: \((\alpha \times \alpha)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-respects-pred \(\text{Rel Pred} \equiv \text{rel-preserves-pred Rel Pred} \land \text{rel-reflects-pred Rel Pred}\)

abbreviation rel-respects-binary-pred :: \((\alpha \times \alpha)\set\Rightarrow (\alpha \Rightarrow \beta) \Rightarrow \text{bool}\)
  where
  rel-respects-binary-pred \(\text{Rel Pred} \equiv \text{rel-preserves-binary-pred Rel Pred} \land \text{rel-reflects-binary-pred Rel Pred}\)

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

lemma symm-relation-impl-preservation-equals-reflection:
  fixes \(\text{Rel} :: (\alpha \times \alpha)\set\)
  \(\text{and Pred :: } \alpha \Rightarrow \beta\)
  \(\text{shows symm: symm Rel}\)
  \(\text{shows rel-preserves-pred Rel Pred = rel-reflects-pred Rel Pred}\)
  \(\text{and rel-preserves-pred Rel Pred = rel-respects-pred Rel Pred}\)
  \(\text{and rel-reflects-pred Rel Pred = rel-respects-pred Rel Pred}\)
    using symm
    unfolding sym-def
    by blast

lemma symm-relation-impl-preservation-equals-reflection-of-binary-predicates:
fixes \( \text{Rel} \) :: \(((a \times a) \text{ set})\) 
and \( \text{Pred} \) :: \('a \Rightarrow 'b \Rightarrow \text{bool}\) 
assumes \text{symm} : \text{sym} \text{Rel} 
shows \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} = \text{rel-reflects-binary-pred} \text{Rel} \text{Pred} 
and \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} = \text{rel-respects-binary-pred} \text{Rel} \text{Pred} 
and \text{rel-reflects-binary-pred} \text{Rel} \text{Pred} = \text{rel-respects-binary-pred} \text{Rel} \text{Pred}
using \text{symm} 
unfolding \text{sym-def} 
by \text{blast}+

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

**Lemma preservation-and-closures:**
fixes \( \text{Rel} \) :: \(((a \times a) \text{ set})\) 
and \( \text{Pred} \) :: \('a \Rightarrow 'b \Rightarrow \text{bool}\) 
assumes \text{preservation} : \text{rel-preserves-pred} \text{Rel} \text{Pred} 
shows \text{rel-preserves-pred} \text{Rel} \text{Pred} = \text{rel-reflects-pred} \text{Rel} \text{Pred} 
and \text{rel-preserves-pred} \text{Rel} \text{Pred} = \text{rel-respects-pred} \text{Rel} \text{Pred} 
and \text{rel-reflects-pred} \text{Rel} \text{Pred} = \text{rel-respects-pred} \text{Rel} \text{Pred}
proof
from \text{preservation} show A : \text{rel-preserves-pred} \text{Rel} \text{Pred}
by (auto simp add: refl)
have B : \( \bigwedge \text{Rel}. \text{rel-preserves-pred} \text{Rel} \text{Pred} \Rightarrow \text{rel-preserves-pred} \text{Rel} \text{Pred}\)
proof clarify
fix \( a \ b \)
assume \((a, b) \in \text{Rel}^+ \) and \text{rel-preserves-pred} \text{Rel} \text{Pred} and \Pred a
thus \Pred b
  by (induct, blast+)
qed
with \text{preservation} show \text{rel-preserves-pred} \text{Rel} \text{Pred}
by blast
from \text{preservation} A B[where \text{Rel}=\text{Rel}^=] show \text{rel-preserves-pred} \text{Rel} \text{Pred}
using \text{trancl-refl}[of \text{Rel}]
by blast
qed

**Lemma preservation-of-binary-predicates-and-closures:**
fixes \( \text{Rel} \) :: \(((a \times a) \text{ set})\) 
and \( \text{Pred} \) :: \('a \Rightarrow 'b \Rightarrow \text{bool}\) 
assumes \text{preservation} : \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} 
shows \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} = \text{rel-reflects-binary-pred} \text{Rel} \text{Pred} 
and \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} = \text{rel-respects-binary-pred} \text{Rel} \text{Pred} 
and \text{rel-reflects-binary-pred} \text{Rel} \text{Pred} = \text{rel-respects-binary-pred} \text{Rel} \text{Pred}
proof
from \text{preservation} show A : \text{rel-preserves-binary-pred} \text{Rel} \text{Pred}
by (auto simp add: refl)
have B : \( \bigwedge \text{Rel}. \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} \Rightarrow \text{rel-preserves-binary-pred} \text{Rel} \text{Pred}\)
proof clarify
fix \( a \ b \ x \)
assume \((a, b) \in \text{Rel}^+ \) and \text{rel-preserves-binary-pred} \text{Rel} \text{Pred} and \Pred a \ x
thus \Pred b \ x
  by (induct, blast+)
qed
with \text{preservation} show \text{rel-preserves-binary-pred} \text{Rel} \text{Pred}
by blast
from \text{preservation} A B[where \text{Rel}=\text{Rel}^=] 
show \text{rel-preserves-binary-pred} \text{Rel} \text{Pred}
using \text{trancl-refl}[of \text{Rel}]
by fast
qed
If a relation reflects a predicate then so does its reflexive or/and transitive closure.

**Lemma reflection-and-closures:**

```plaintext
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
  from reflection show A: rel-reflects-pred (Rel^=) Pred
    by (auto simp add: refl)
  have B: ∨ Rel. rel-reflects-pred Rel Pred ⇒ rel-reflects-pred (Rel^+) Pred
proof clarify
  fix Rel a b
  assume (a, b) ∈ Rel^+ and rel-reflects-pred Rel Pred and Pred b
  thus Pred a
    by (induct, blast+)
qed
  with reflection show rel-reflects-pred (Rel^+) Pred
    by blast
  from reflection A B[where Rel=Rel^=]
  show rel-reflects-pred (Rel^*) Pred
    using trancl-reflcl[of Rel]
    by fast
qed
```

**Lemma reflection-of-binary-predicates-and-closures:**

```plaintext
fixes Rel :: ('a × 'a) set
  and Pred :: 'a ⇒ 'b ⇒ bool
assumes reflection: rel-reflects-binary-pred Rel Pred
shows rel-reflects-binary-pred (Rel^=) Pred
  and rel-reflects-binary-pred (Rel^+) Pred
  and rel-reflects-binary-pred (Rel^*) Pred
proof
  from reflection show A: rel-reflects-binary-pred (Rel^=) Pred
    by (auto simp add: refl)
  have B: ∨ Rel. rel-reflects-binary-pred Rel Pred ⇒ rel-reflects-binary-pred (Rel^+) Pred
proof clarify
  fix Rel a b x
  assume (a, b) ∈ Rel^+ and rel-reflects-binary-pred Rel Pred and Pred b x
  thus Pred a x
    by (induct, blast+)
qed
  with reflection show rel-reflects-binary-pred (Rel^+) Pred
    by blast
  from reflection A B[where Rel=Rel^=]
  show rel-reflects-binary-pred (Rel^*) Pred
    using trancl-reflcl[of Rel]
    by fast
qed
```

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

**Lemma respection-and-closures:**

```plaintext
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
proof
  from reflection show A: rel-respects-pred (Rel^=) Pred
    by (auto simp add: refl)
  have B: ∨ Rel. rel-respects-pred Rel Pred ⇒ rel-respects-pred (Rel^+) Pred
proof clarify
  fix Rel a b x
  assume (a, b) ∈ Rel^+ and rel-respects-pred Rel Pred and Pred b x
  thus Pred a x
    by (induct, blast+)
qed
  with reflection show rel-respects-pred (Rel^+) Pred
    by blast
  from reflection A B[where Rel=Rel^=]
  show rel-respects-pred (Rel^*) Pred
    using trancl-reflcl[of Rel]
    by fast
qed
```
and rel-respects-pred \((symcl (Rel^=))^+\) Pred

proof
from respection show A: rel-respects-pred (Rel^=) Pred
  using preservation-and-closures(1)[where Rel=Rel and Pred=Pred]
  reflection-and-closures(1)[where Rel=Rel and Pred=Pred]
  by blast
have B: \(\forall Rel. \rel-respects-pred Rel Pred \implies \rel-respects-pred (symcl Rel) Pred\)
proof
  fix Rel
  assume B1: rel-respects-pred Rel Pred
  show rel-preserves-pred (symcl Rel) Pred
    proof clarify
      fix a b
      assume (a, b) \(\in\) symcl Rel
      hence (a, b) \(\in\) Rel \(\lor\) (b, a) \(\in\) Rel
      by (simp add: symcl-def)
      moreover assume Pred a
      ultimately show Pred b
        using B1
      by blast
    qed
  qed
next
  fix Rel :: (\('a \times 'a\) set)
  and Pred :: 'a \(\Rightarrow\) bool
  assume B2: rel-respects-pred Rel Pred
  show rel-reflects-pred (symcl Rel) Pred
    proof clarify
      fix a b
      assume (a, b) \(\in\) symcl Rel
      hence (a, b) \(\in\) Rel \(\lor\) (b, a) \(\in\) Rel
      by (simp add: symcl-def)
      moreover assume Pred b
      ultimately show Pred a
        using B2
      by blast
    qed
  qed
from respection B[where Rel=Rel] show rel-respects-pred (symcl Rel) Pred by blast
have C: \(\forall Rel. \rel-respects-pred Rel Pred \implies \rel-respects-pred (Rel^+) Pred\)
proof
  fix Rel
  assume rel-respects-pred Rel Pred
  thus rel-respects-pred (Rel^+) Pred
    using preservation-and-closures(2)[where Rel=Rel and Pred=Pred]
    reflection-and-closures(2)[where Rel=Rel and Pred=Pred]
    by blast
  qed
from respection C[where Rel=Rel] show rel-respects-pred (Rel^+) Pred by blast
from A B[where Rel=Rel^=] show rel-respects-pred (symcl (Rel^=)) Pred by blast
from A C[where Rel=Rel^=] show rel-respects-pred (Rel^+) Pred
  using trancl-reflcl[of Rel]
  by fast
from A B[where Rel=Rel^=] C[where Rel=symcl (Rel^=)] show rel-respects-pred ((symcl (Rel^=))^+) Pred
  by blast
qed

lemma respection-of-binary-predicates-and-closures:
fixes Rel :: ('a × '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 (symcl (Rel')) Pred

proof –
  from respection show A: rel-respects-binary-pred (Rel') Pred
    using preservation-of-binary-predicates-and-closures(1)[where Rel=Rel and Pred=Pred]
    reflection-of-binary-predicates-and-closures(1)[where Rel=Rel and Pred=Pred]
  by blast
have B: ⋀Rel. rel-respects-binary-pred Rel Pred ⇒ rel-respects-binary-pred (symcl Rel) Pred
proof
  fix Rel
  assume B1: rel-respects-binary-pred Rel Pred
  show rel-preserves-binary-pred (symcl Rel) Pred
  proof clarify
    fix a b x
    assume (a, b) ∈ symcl Rel
    hence (a, b) ∈ Rel ∨ (b, a) ∈ Rel
      by (simp add: symcl-def)
    moreover assume Pred a x
    ultimately show Pred b x
      using B1
    by blast
  qed
next
  fix Rel
  assume B2: rel-respects-binary-pred Rel Pred
  show rel-reflects-binary-pred (symcl Rel) Pred
  proof clarify
    fix a b x
    assume (a, b) ∈ symcl Rel
    hence (a, b) ∈ Rel ∨ (b, a) ∈ Rel
      by (simp add: symcl-def)
    moreover assume Pred b x
    ultimately show Pred a x
      using B2
    by blast
  qed
qed

from respection B[where Rel=Rel] show rel-respects-binary-pred (symcl Rel) Pred
  by blast
have C: ⋀Rel. rel-respects-binary-pred Rel Pred ⇒ rel-respects-binary-pred (Rel') Pred
proof –
  fix Rel
  assume rel-respects-binary-pred Rel Pred
  thus rel-respects-binary-pred (Rel') Pred
    using preservation-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]
    reflection-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]
  by blast
qed

from respection C[where Rel=Rel] show rel-respects-binary-pred (Rel') Pred
  by blast
from A B[where Rel=Rel']
  show rel-respects-binary-pred (symcl (Rel')) Pred
  by blast
from A C[where Rel=Rel']
2 Process Calculi

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

2.1 Reduction Semantics

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

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

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

abbreviation step :: ‘proc ⇒ ‘proc processCalculus ⇒ ‘proc ⇒ bool
  (-⇒- - [70, 70, 70] 80)
  where
  P ⇒⇒ Cal Q ≡ Reductions Cal P Q

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

primrec nSteps
  :: ‘proc ⇒ ‘proc processCalculus ⇒ nat ⇒ ‘proc ⇒ bool
  (-⇒- - [70, 70, 70] 80)
  where
  P ⇒⇒ Cal^0 Q = (P = Q) |
  P ⇒⇒ Cal^Suc n Q = (∃ P’. P ⇒⇒ Cal^n P’ ∧ P’ ⇒⇒ Cal Q)

definition steps
  :: ‘proc ⇒ ‘proc processCalculus ⇒ ‘proc ⇒ bool
  (-⇒- - [70, 70, 70] 80)
  where
  P ⇒⇒ Cal* Q ≡ ∃ n. P ⇒⇒ Cal^n Q

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

definition divergent
  :: ‘proc ⇒ ‘proc processCalculus ⇒ bool
  (-⇒ω [70, 70] 80)
  where
  P ⇒⇒(Cal)ω ≡ ∀ P’. P ⇒⇒ Cal* P’ ⇒⇒ (∃ P”. P’ ⇒⇒ Cal P”)

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

lemma steps-refl:
  fixes Cal :: ‘proc processCalculus
  and P :: ‘proc
  shows P ⇒⇒ Cal* P
proof
have \( P \mapsto \rightarrow \text{Cal}^0 P \)
  by simp
hence \( \exists n. P \mapsto \rightarrow \text{Cal}^n P \)
  by blast
thus \( P \mapsto \rightarrow \text{Cal}^* P \)
  by (simp add: steps-def)
qed

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

lemma step-to-steps:
  fixes \( \text{Cal} :: \text{'proc processCalculus} \)
  and \( P P' :: \text{'proc} \)
  assumes step: \( P \mapsto \rightarrow \text{Cal} P' \)
  shows \( P \mapsto \rightarrow \text{Cal}^* P' \)
proof
  from step have \( P \mapsto \rightarrow \text{Cal}^1 P' \)
  by simp
  thus \?thesis
    unfolding steps-def
    by blast
qed

If there is a sequence of steps from \( P \) to \( Q \) and from \( Q \) to \( R \), then there is also a sequence of steps from \( P \) to \( R \).

lemma nSteps-add:
  fixes \( \text{Cal} :: \text{'proc processCalculus} \)
  and \( n1 n2 :: \text{nat} \)
  shows \( \forall P Q R. P \mapsto \rightarrow \text{Cal}^{n1} Q \land Q \mapsto \rightarrow \text{Cal}^{n2} R \mapsto \rightarrow P \mapsto \rightarrow \text{Cal}^{(n1 + n2)} R \)
proof (induct n2, simp)
case (Suc n)
  assume IH: \( \forall P Q R. P \mapsto \rightarrow \text{Cal}^{n1} Q \land Q \mapsto \rightarrow \text{Cal}^n R \mapsto \rightarrow P \mapsto \rightarrow \text{Cal}^{(n1 + n) R} \)
  show ?case
    proof clarify
      fix \( P Q R \)
      assume \( Q \mapsto \rightarrow \text{Cal}^{\text{Suc } n} R \)
      from this obtain \( Q' \) where \( A1: Q \mapsto \rightarrow \text{Cal}^{n1} Q' \) and \( A2: Q' \mapsto \rightarrow \text{Cal}^n R \)
        by auto
      assume \( P \mapsto \rightarrow \text{Cal}^{n1} Q \)
      with \( A1 \) IH have \( P \mapsto \rightarrow \text{Cal}^{(n1 + n)} Q' \)
        by blast
      with \( A2 \) show \( P \mapsto \rightarrow \text{Cal}^{(n1 + \text{Suc } n)} R \)
        by auto
    qed
qed

lemma steps-add:
  fixes \( \text{Cal} :: \text{'proc processCalculus} \)
  and \( P Q R :: \text{'proc} \)
  assumes \( A1: P \mapsto \rightarrow \text{Cal}^* Q \)
  and \( A2: Q \mapsto \rightarrow \text{Cal}^* R \)
  shows \( P \mapsto \rightarrow \text{Cal}^* R \)
proof
  from \( A1 \) obtain \( n1 \) where \( P \mapsto \rightarrow \text{Cal}^{n1} Q \)
    by (auto simp add: steps-def)
  moreover from \( A2 \) obtain \( n2 \) where \( Q \mapsto \rightarrow \text{Cal}^{n2} R \)
    by (auto simp add: steps-def)
  ultimately have \( P \mapsto \rightarrow \text{Cal}^{(n1 + n2)} R \)
    using nSteps-add[where \( \text{Cal} = \text{Cal} \)]
    by blast
qed
thus \( P \mapsto \text{Cal}^* \ R \)
by (simp add: steps-def, blast)
qed

2.1.1 Observables or Barbs

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

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

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

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

abbreviation reachesBarb
:: ('proc ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs ⇒ bool
  (¬↓<- - [70, 70] 80)
where
P ↓< CWB > a ≡ (∃P'. P ⇒¬ (Calculus CWB)* P' ∧ P' ↓< CWB > a)

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

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

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

lemma preservation-of-barbs-and-set-of-barbs:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  shows rel-preserves-barbs-barb-set Rel CWB = (∀Barbs. rel-preserves-barb-set Rel CWB Barbs)
  by blast

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

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

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

lemma reflection-of-barbs-and-set-of-barbs:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  shows rel-reflects-barbs-barb-set Rel CWB = (∀Barbs. rel-reflects-barb-set Rel CWB Barbs)
A relation respects barbs if it preserves and reflects barbs.

**abbreviation** rel-respects-barb-set
:: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ 'barbs set ⇒ bool

where
rel-respects-barb-set Rel CWB Barbs ≡
rel-preserves-barb-set Rel CWB Barbs ∧ rel-reflects-barb-set Rel CWB Barbs

**abbreviation** rel-respects-barbs
:: ('proc × 'proc) set ⇒ ('proc, 'barbs) calculusWithBarbs ⇒ bool

where
rel-respects-barbs Rel CWB ≡ rel-preserves-barbs Rel CWB ∧ rel-reflects-barbs Rel CWB

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

*fixes* Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
*shows* rel-respects-barbs Rel CWB = (∀ Barbs. rel-respects-barb-set Rel CWB Barbs)
  by blast

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

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

*fixes* Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
*assumes* preservation: rel-preserves-barbs Rel CWB
*shows* rel-preserves-barbs (Rel−) CWB
  and rel-preserves-barbs (Rel+) CWB
  and rel-preserves-barbs (Rel∗) CWB
  using preservation
    preservation-of-binary-predicates-and-closures[where Rel=Rel and Pred=HasBarb CWB]
  by blast

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

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

*fixes* Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
*assumes* reflection: rel-reflects-barbs Rel CWB
*shows* rel-reflects-barbs (Rel−) CWB
  and rel-reflects-barbs (Rel+) CWB
  and rel-reflects-barbs (Rel∗) CWB
  using reflection
    reflection-of-binary-predicates-and-closures[where Rel=Rel and Pred=HasBarb CWB]
  by blast

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

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

*fixes* Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
*assumes* respection: rel-respects-barbs Rel CWB
*shows* rel-respects-barbs (Rel+) CWB
  and rel-respects-barbs (symcl Rel) CWB
  and rel-respects-barbs (Rel+) CWB
  and rel-respects-barbs (symcl (Rel−)) CWB
  and rel-respects-barbs (Rel∗) CWB
  and rel-respects-barbs ((symcl (Rel−))+) CWB
*proof* –
  from respection show rel-respects-barbs (Rel+) CWB
    using respection-of-binary-predicates-and-closures[1][where Rel=Rel and Pred=HasBarb CWB]
    by blast
  next
from respection show rel-respects-barbs (symcl Rel) CWB
using respection-of-binary-predicates-and-closures[2][where Rel=Rel and Pred=HasBarb CWB]
by blast
next
from respection show rel-respects-barbs (Rel+) CWB
using respection-of-binary-predicates-and-closures[3][where Rel=Rel and Pred=HasBarb CWB]
by blast
next
from respection show rel-respects-barbs (symcl (Rel+)) CWB
using respection-of-binary-predicates-and-closures[4][where Rel=Rel and Pred=HasBarb CWB]
by blast
next
from respection show rel-respects-barbs (symcl (Rel*)) CWB
using respection-of-binary-predicates-and-closures[5][where Rel=Rel and Pred=HasBarb CWB]
by blast
next
from respection show rel-respects-barbs ((symcl (Rel*))+) CWB
using respection-of-binary-predicates-and-closures[6][where Rel=Rel and Pred=HasBarb CWB]
by blast
qed

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

abbreviation rel-weakly-preserves-barb-set
ː (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ ′barbs set ⇒ bool
where
  rel-weakly-preserves-barb-set Rel CWB Barbs ≡
  rel-preserves-binary-pred Rel (λP a. a ∈ Barbs ∧ P⇓<CWB>a)
abbreviation rel-weakly-preserves-barbs
ː (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ bool
where
  rel-weakly-preserves-barbs Rel CWB ≡ rel-preserves-binary-pred Rel (λP a. P⇓<CWB>a)

lemma weak-preservation-of-barbs-and-set-of-barbs:
fixes Rel :: (′proc × ′proc) set
and CWB :: (′proc, ′barbs) calculusWithBarbs
shows rel-weakly-preserves-barbs Rel CWB
  = (∀ Barbs. rel-weakly-preserves-barb-set Rel CWB Barbs)
by blast

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

abbreviation rel-weakly-reflects-barb-set
ː (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ ′barbs set ⇒ bool
where
  rel-weakly-reflects-barb-set Rel CWB Barbs ≡
  rel-reflects-binary-pred Rel (λP a. a ∈ Barbs ∧ P⇓<CWB>a)
abbreviation rel-weakly-reflects-barbs
ː (′proc × ′proc) set ⇒ (′proc, ′barbs) calculusWithBarbs ⇒ bool
where
  rel-weakly-reflects-barbs Rel CWB ≡ rel-reflects-binary-pred Rel (λP a. P⇓<CWB>a)

lemma weak-reflection-of-barbs-and-set-of-barbs:
fixes Rel :: (′proc × ′proc) set
and CWB :: (′proc, ′barbs) calculusWithBarbs
shows rel-weakly-reflects-barbs Rel CWB = (∀ Barbs. rel-weakly-reflects-barb-set Rel CWB Barbs)
by blast

A relation weakly respects barbs if it weakly preserves and weakly reflects barbs.
abbreviation \text{rel-weakly-respects-barb-set} :: (\text{\texttt{proc}} \times \text{\texttt{proc}}) \text{set} \Rightarrow (\text{\texttt{proc}}, \text{\texttt{barbs}}) \text{calculusWithBarbs} \Rightarrow \text{\texttt{barbs set}} \Rightarrow \text{bool}

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

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

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

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

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

lemma \text{weak-preservation-of-barbs-and-closures}:
\text{fixes Rel} :: (\text{\texttt{proc}} \times \text{\texttt{proc}}) \text{set}
and \text{CWB} :: (\text{\texttt{proc}}, \text{\texttt{barbs}}) \text{calculusWithBarbs}
assumes \text{preservation: rel-weakly-preserves-barbs Rel CWB}
shows \text{rel-weakly-preserves-barbs (Rel\textsuperscript{\texttt{\footnotesize +}}) CWB}
and \text{rel-weakly-preserves-barbs (Rel\textsuperscript{\texttt{\footnotesize *}}) CWB}
and \text{rel-weakly-preserves-barbs (symcl (Rel\textsuperscript{\texttt{\footnotesize =}})) CWB}
using \text{preservation preservation-of-binary-predicates-and-closures[where Rel=Rel}}
and \text{Pred=\lambda P a. P\downarrow<\text{CWB}>a]}
\text{by blast}$\mathbf{+}$

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

lemma \text{weak-reflection-of-barbs-and-closures}:
\text{fixes Rel} :: (\text{\texttt{proc}} \times \text{\texttt{proc}}) \text{set}
and \text{CWB} :: (\text{\texttt{proc}}, \text{\texttt{barbs}}) \text{calculusWithBarbs}
assumes \text{reflection: rel-weakly-reflects-barbs Rel CWB}
shows \text{rel-weakly-reflects-barbs (Rel\textsuperscript{\texttt{\footnotesize =}}) CWB}
and \text{rel-weakly-reflects-barbs (Rel\textsuperscript{\texttt{\footnotesize +}}) CWB}
and \text{rel-weakly-reflects-barbs (symcl (Rel\textsuperscript{\texttt{\footnotesize =}})) CWB}
and \text{rel-weakly-reflects-barbs (\text{\texttt{\footnotesize #}}(\text{\texttt{\footnotesize symcl (Rel\textsuperscript{\texttt{\footnotesize =}})})) CWB)
using \text{reflection reflection-of-binary-predicates-and-closures[where Rel=Rel}}
and \text{Pred=\lambda P a. P\downarrow<\text{CWB}>a]}
\text{by blast}$\mathbf{+}$

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

lemma \text{weak-respection-of-barbs-and-closures}:
\text{fixes Rel} :: (\text{\texttt{proc}} \times \text{\texttt{proc}}) \text{set}
and \text{CWB} :: (\text{\texttt{proc}}, \text{\texttt{barbs}}) \text{calculusWithBarbs}
assumes \text{respection: rel-weakly-respects-barbs Rel CWB}
shows \text{rel-weakly-respects-barbs (Rel\textsuperscript{\texttt{\footnotesize =}}) CWB}
and \text{rel-weakly-respects-barbs (symcl Rel) CWB}
and \text{rel-weakly-respects-barbs (Rel\textsuperscript{\texttt{\footnotesize +}}) CWB}
and \text{rel-weakly-respects-barbs (\text{\texttt{\footnotesize #}}(\text{\texttt{\footnotesize symcl (Rel\textsuperscript{\texttt{\footnotesize =}})}) CWB)
proof
from \text{respection show rel-weakly-respects-barbs (Rel\textsuperscript{\texttt{\footnotesize =}}) CWB}
using \text{respection-of-binary-predicates-and-closures(1)[where Rel=Rel}}
and \text{Pred=\lambda P a. P\downarrow<\text{CWB}>a]}
\text{by blast}
\text{next}
from \text{respection show rel-weakly-respects-barbs (symcl Rel) CWB}
using respection-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=λP a. P↓<CWB>a]
by blast
next
from respection show rel-weakly-respects-barbs (Rel+) CWB
using respection-of-binary-predicates-and-closures(3)[where Rel=Rel and Pred=λP a. P↓<CWB>a]
by blast
next
from respection show rel-weakly-respects-barbs (symcl (Rel=)) CWB
using respection-of-binary-predicates-and-closures(4)[where Rel=Rel and Pred=λP a. P↓<CWB>a]
by blast
next
from respection show rel-weakly-respects-barbs (symcl (Rel=)) CWB
using respection-of-binary-predicates-and-closures(5)[where Rel=Rel and Pred=λP a. P↓<CWB>a]
by blast
next
from respection show rel-weakly-respects-barbs (symcl (Rel=)) CWB
using respection-of-binary-predicates-and-closures(6)[where Rel=Rel and Pred=λP a. P↓<CWB>a]
by blast
qed

end
theory SimulationRelations
imports ProcessCalculi
begin

3 Simulation Relations

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

3.1 Simulation

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

abbreviation weak-reduction-simulation
:: (‘proc × ‘proc) set ⇒ ‘proc processCalculus ⇒ bool
where
weak-reduction-simulation Rel Cal ≡
∀ P Q P’. (P, Q) ∈ Rel ∧ P ↦−→ Cal∗ P’ ↦−→ (∃ Q’. Q ↦−→ Cal∗ Q’ ∧ (P’, Q’) ∈ Rel)

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

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

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

lemma weak-reduction-simulation-and-closures:
fixes Rel :: (‘proc × ‘proc) set
and Cal :: ‘proc processCalculus
assumes simulation: weak-reduction-simulation Rel Cal
proof
from simulation show A: weak-reduction-simulation \( (\text{Rel}^=) \) Cal
by (auto simp add: refl, blast)
have B: \( \land \text{Rel. weak-reduction-simulation Rel Cal} \implies \text{weak-reduction-simulation (Rel}^+) \) Cal
proof clarify
fix Rel \( P \) \( Q \) \( P' \)
assume B1: weak-reduction-simulation Rel Cal
assume \( (P, Q) \in \text{Rel}^+ \) and \( P \mapsto \text{Cal}^* \) \( P' \)
thus \( \exists Q'. Q \mapsto \text{Cal}^* \) \( Q' \land (P', Q') \in \text{Rel}^+ \)
proof (induct arbitrary: \( P' \))
fix Q \( P' \)
assume \( (P, Q) \in \text{Rel} \) and \( P \mapsto \text{Cal}^* \) \( P' \)
with B1 obtain \( Q' \) where \( Q \mapsto \text{Cal}^* \) \( Q' \land (P', Q') \in \text{Rel}^+ \)
by blast
thus \( \exists Q'. Q \mapsto \text{Cal}^* \) \( Q' \land (P', Q') \in \text{Rel}^+ \)
by auto
next
case (step \( Q \) \( R \) \( P' \))
assume \( \land \) \( P', P \mapsto \text{Cal}^* \) \( P' \implies (\exists Q'. Q \mapsto \text{Cal}^* \) \( Q' \land (P', Q') \in \text{Rel}^+ \)
and \( P \mapsto \text{Cal}^* \) \( P' \)
from this obtain \( Q' \) where B2: \( Q \mapsto \text{Cal}^* \) \( Q' \land (P', Q') \in \text{Rel}^+ \)
by blast
assume \( (Q, R) \in \text{Rel} \)
with B1 B2 obtain \( R' \) where B4: \( R \mapsto \text{Cal}^* \) \( R' \land (P', R') \in \text{Rel}^+ \)
by simp
from B4 this show \( \exists R'. R \mapsto \text{Cal}^* \) \( R' \land (P', R') \in \text{Rel}^+ \)
by blast
qed
qed

with simulation show weak-reduction-simulation (\( \text{Rel}^+ \)) Cal
by blast
from simulation A B[where \( \text{Rel}=\text{Rel}^= \)]
show weak-reduction-simulation (\( \text{Rel}^* \)) Cal
using trancl-refcl[of Rel]
by fast
qed

lemma weak-barbed-simulation-and-closures:
fixes \( \text{Rel} :: (\text{proc } \times \text{proc} ) \text{ set} \)
and \( \text{CWB} :: (\text{proc } , \text{barbs} ) \text{ calculusWithBarbs} \)
assumes simulation: weak-barbed-simulation \( \text{Rel} \) \( \text{CWB} \)
shows weak-barbed-simulation (\( \text{Rel}^= \)) \( \text{CWB} \)
and weak-barbed-simulation (\( \text{Rel}^+ \)) \( \text{CWB} \)
and weak-barbed-simulation (\( \text{Rel}^* \)) \( \text{CWB} \)
proof
from simulation show weak-barbed-simulation (\( \text{Rel}^= \)) \( \text{CWB} \)
using weak-reduction-simulation-and-closures(1)[where \( \text{Rel}=\text{Rel} \) and \( \text{Cal}=\text{Calculus CWB} \)]
weak-preservation-of-barbs-and-closures(1)[where \( \text{Rel}=\text{Rel} \) and \( \text{CWB}=\text{CWB} \)]
by blast
next
from simulation show weak-barbed-simulation (\( \text{Rel}^+ \)) \( \text{CWB} \)
using weak-reduction-simulation-and-closures(2)[where \( \text{Rel}=\text{Rel} \) and \( \text{Cal}=\text{Calculus CWB} \)]
weak-preservation-of-barbs-and-closures(2)[where \( \text{Rel}=\text{Rel} \) and \( \text{CWB}=\text{CWB} \)]
by blast
next
from simulation show weak-barbed-simulation (\( \text{Rel}^* \)) \( \text{CWB} \)
using weak-reduction-simulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
weak-preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]

by blast
qed

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

abbreviation weak-barbed-preservation-cond :: \((\text{'proc } \times \text{'proc})\) set ⇒ \((\text{'proc }, \text{'barbs})\) calculusWithBarbs ⇒ bool

where
weak-barbed-preservation-cond Rel CWB ≡ ∀ P Q a. \((P, Q)\) ∈ Rel ∧ \(P\downarrow\downarrow CWB > a\) \(\rightarrow\) \(Q\downarrow\downarrow CWB > a\)

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

proof clarify
fix P Q a
have P ↦−→(Calculus CWB)* P
by (simp add: steps-refl)
moreover assume \(P\downarrow\downarrow CWB > a\)
ultimately have P\downarrow\downarrow CWB > a
by blast
moreover assume \((P, Q)\) ∈ Rel
ultimately show Q\downarrow\downarrow CWB > a
using preservation
by blast
qed

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

proof
assume rel-weakly-preserves-barbs Rel CWB
thus weak-barbed-preservation-cond Rel CWB
using weak-preservation-of-barbs[where Rel=Rel and CWB=CWB]
by blast
next
assume condition: weak-barbed-preservation-cond Rel CWB
show rel-weakly-preserves-barbs Rel CWB

proof clarify
fix P Q a P'
assume \((P, Q)\) ∈ Rel and \(P\) ↦→(Calculus CWB)* P'
with simulation obtain Q' where A1: \(Q\) ↦→(Calculus CWB)* Q' and A2: \((P', Q')\) ∈ Rel
by blast
assume \(P'\downarrow\downarrow CWB > a\)
with A2 condition obtain Q'' where A3: \(Q'\) ↦→(Calculus CWB)* Q'' and A4: \(Q''\downarrow\downarrow CWB > a\)
by blast
from A1 A3 have \(Q\) ↦→(Calculus CWB)* Q''
by (rule steps-add)
with A4 show Q\downarrow\downarrow CWB > a
by blast
qed
qed

A strong reduction simulation is relation \(R\) such that for each pair \((P, Q)\) in \(R\) and each step of \(P\) to some \(P'\) there exists some \(Q'\) such that there is a step of \(Q\) to \(Q'\) and \((P', Q')\) in \(R\).
abbreviation strong-reduction-simulation :: ('proc × 'proc) set ⇒ 'proc processCalculus ⇒ bool
where
strong-reduction-simulation Rel Cal ≡
∀ P Q P’. (P, Q) ∈ Rel ∧ P −→ Cal P’ −→ (∃ Q’. Q −→ Cal Q’ ∧ (P’, Q’) ∈ Rel)

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

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

A strong strong simulation is also a weak simulation.

lemma strong-impl-weak-reduction-simulation:
  fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
  assumes simulation: strong-reduction-simulation Rel Cal
  shows weak-reduction-simulation Rel Cal
proof clarify
  fix P Q P’
  assume A1: (P, Q) ∈ Rel
  assume P −→ Cal∗ P’
  from this obtain n where P −→ Calⁿ P’
  by (auto simp add: steps-def)
  thus ∃ Q’. Q −→ Cal∗ Q’ ∧ (P’, Q’) ∈ Rel
proof (induct n arbitrary: P’)
  case 0
  assume P −→ Cal⁰ P’
  hence P = P’
  by (simp add: steps-refl)
  moreover have Q −→ Cal∗ Q
  by (rule steps-refl)
  ultimately show ∃ Q’. Q −→ Cal∗ Q’ ∧ (P’, Q’) ∈ Rel
    using A1
  by blast
next
  case (Suc n P’)
  assume P −→ Calⁿ P’
  from this obtain P’ where A2: P −→ Calⁿ P’ and A3: P’ −→ Cal P’n
  by auto
  assume P’ P −→ Calⁿ P’ −→ ∃ Q’. Q −→ Cal∗ Q’ ∧ (P’, Q’) ∈ Rel
  with A2 obtain Q’ where A4: Q −→ Cal∗ Q’ and A5: (P’, Q’) ∈ Rel
  by blast
  from simulation A5 A3 obtain Q” where A6: Q’ −→ Cal Q” and A7: (P”, Q”) ∈ Rel
  by blast
  from A4 A6 have Q −→ Cal∗ Q”
    using steps-add[where P = Q and Q = Q’ and R = Q”]
  by (simp add: step-to-steps)
  with A7 show ∃ Q’. Q −→ Cal∗ Q’ ∧ (P”, Q”) ∈ Rel
  by blast
qed

lemma strong-barbed-simulation-impl-weak-preservation-of-barbs:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes simulation: strong-barbed-simulation Rel CWB
  shows rel-weakly-preserves-barbs Rel CWB
proof clarify
  fix P Q a P’
lemma strong-reduction-simulation-and-closures:

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

assumes simulation: strong-reduction-simulation \( Rel \) \( \text{CWB} \)

shows weak-barbed-simulation \( Rel \) \( \text{CWB} \)

using simulation

\( \Rightarrow \) strong-reduction-simulation \( Rel \) \( \text{CWB} \)

by blast

proof

from simulation show A: strong-reduction-simulation \( (Rel^+) \) \( Cal \)

by (auto simp add: refl, blast)

have B: \( \forall \text{Rel}. \) strong-reduction-simulation \( Rel \) \( Cal \)

\( \Rightarrow \) strong-reduction-simulation \( (Rel^+) \) \( Cal \)

proof clarify

fix \( Rel \) \( P \) \( Q \) \( P' \)

assume B1: strong-reduction-simulation \( Rel \) \( Cal \)

assume \( (P, Q) \in Rel^+ \) and \( P \mapsto Cal P' \)

thus \( \exists Q'. \ P \mapsto Cal Q' \land (P', Q') \in Rel^+ \)

proof (induct arbitrary: \( P' \))

fix \( Q \) \( P' \)

assume \( (P, Q) \in Rel \) and \( P \mapsto Cal P' \)

with B1 obtain \( Q' \) where \( Q \mapsto Cal Q' \land (P', Q') \in Rel \)

by blast

thus \( \exists Q'. \ P \mapsto Cal Q' \land (P', Q') \in Rel^+ \)

by auto

next

case \( (step \ Q \ R \ P') \)

assume \( \forall P'. \ P \mapsto Cal P' \Rightarrow (\exists Q'. \ Q \mapsto Cal Q' \land (P', Q') \in Rel^+) \)

and \( P \mapsto Cal P' \)

from this obtain \( Q' \) where B2: \( Q \mapsto Cal Q' \land B3: (P', Q') \in Rel^+ \)

by blast

assume \( (Q, R) \in Rel \)

with B1 B2 obtain \( R' \) where B4: \( R \mapsto Cal R' \land B5: (Q', R') \in Rel^+ \)

by blast

from B3 B5 have \( (P', R') \in Rel^+ \)

by simp

with B4 show \( \exists R'. \ R \mapsto Cal R' \land (P', R') \in Rel^+ \)

by blast

qed

qed
with simulation show strong-reduction-simulation (Rel⁺) Cal
by blast
from simulation A B[where Rel=Relᵐ]
show strong-reduction-simulation (Rel⁺) Cal
  using trancl-refl[of Rel]
  by fast
qed

lemma strong-barbed-simulation-and-closures:
  fixes Rel :: (proc × proc) set
  and CWB :: (proc, barbs) calculusWithBarbs
  assumes simulation: strong-barbed-simulation Rel CWB
  shows strong-barbed-simulation (Rel⁺) CWB
  and strong-barbed-simulation (Rel⁺) CWB
  and strong-barbed-simulation (Rel⁺) CWB
proof
  from simulation show strong-barbed-simulation (Rel⁺) CWB
  using strong-reduction-simulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
  preservation-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
  by blast
next
  from simulation show strong-barbed-simulation (Rel⁺) CWB
  using strong-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
  preservation-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
  by blast
next
  from simulation show strong-barbed-simulation (Rel⁺) CWB
  using strong-reduction-simulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
  preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
  by blast
qed

3.2 Contrasimulation

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

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

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

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

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

lemma weak-reduction-contrasimulation-and-closures:
  fixes Rel :: (proc × proc) set
  and Cal :: proc processCalculus
  assumes contrasimulation: weak-reduction-contrasimulation Rel Cal
  shows weak-reduction-contrasimulation (Rel⁺) Cal
  and weak-reduction-contrasimulation (Rel⁺) Cal
  and weak-reduction-contrasimulation (Rel⁺) Cal
proof
  from contrasimulation show A: weak-reduction-contrasimulation (Rel⁺) Cal
lemma weak-barbed-contrasimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes contrasimulation: weak-barbed-contrasimulation Rel CWB
  shows weak-barbed-contrasimulation (Rel^+) CWB
  and weak-barbed-contrasimulation (Rel^) CWB
proof ~
  from contrasimulation show weak-barbed-contrasimulation (Rel^+) CWB
    using weak-reduction-contrasimulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
    weak-preservation-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
    by blast
next
  from contrasimulation show weak-barbed-contrasimulation (Rel^+) CWB
    using weak-reduction-contrasimulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
    weak-preservation-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
    by blast
next
  from contrasimulation show weak-barbed-contrasimulation (Rel^) CWB
    using weak-reduction-contrasimulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
    weak-preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
    by blast
qed
3.3 Coupled Simulation

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

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

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

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

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

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

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

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

lemma weak-reduction-coupled-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes coupledSimulation: weak-reduction-coupled-simulation Rel Cal
shows weak-reduction-coupled-simulation (Rel+) Cal
using weak-reduction-simulation-and-closures[where Rel=Rel and Cal=Cal]
and weak-reduction-contrasimulation-and-closures[where Rel=Rel and Cal=Cal]
and weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation[where Rel=Rel and Cal=Cal]
coupledSimulation
by auto

lemma weak-barbed-coupled-simulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes coupledSimulation: weak-barred-coupled-simulation Rel CWB
shows weak-barred-coupled-simulation (Rel+) CWB
and weak-barred-coupled-simulation (Rel+) CWB
and weak-barred-coupled-simulation (Rel+) CWB
proof
from coupledSimulation show weak-barred-coupled-simulation (Rel+) CWB
using weak-reduction-coupled-simulation-and-closures(1)[where Rel=Rel

}
and \( \text{Cal} = \text{Calculus CWB} \)
weak-preservation-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
by blast

next
from coupledSimulation show weak-barbed-coupled-simulation \((Rel^+)\) CWB
using weak-reduction-coupled-simulation-and-closures(2)[where Rel=Rel
and \( \text{Cal} = \text{Calculus CWB} \)]
weak-preservation-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
by blast
next
from coupledSimulation show weak-barbed-coupled-simulation \((Rel^*)\) CWB
using weak-reduction-coupled-simulation-and-closures(3)[where Rel=Rel
and \( \text{Cal} = \text{Calculus CWB} \)]
weak-preservation-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
by blast
qed

3.4 Correspondence Simulation

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

abbreviation weak-reduction-correspondence-simulation
\[\begin{align*}
\forall P \times Q \times P'. (P, Q) \in R & \land P \rightarrow \text{Cal} \times P' \rightarrow (Q Q'). Q \rightarrow \text{Cal} \times Q' \land (P', Q') \in R) \\
\land \forall P \times Q Q'. (P, Q) \in R & \land Q \rightarrow \text{Cal} \times Q' \\
\rightarrow (Q Q'). P \rightarrow \text{Cal} \times P' \land Q' \rightarrow \text{Cal} \times Q' \land (P'', Q'') \in R)\end{align*}\] for each weak correspondence simulation \( R \) there exists a weak coupled simulation that contains all pairs of \( R \) in both directions.

inductive-set eSim-cs : (proc \( \times \) proc set \( \Rightarrow \) proc processCalculus \( \Rightarrow \) (proc \( \times \) proc set))
for Rel : (proc \( \times \) proc set)
and \( \text{Cal} : \text{proc processCalculus} \)
where
left: \( [Q \rightarrow \text{Cal} \times Q'; (P', Q') \in R] \rightarrow (P', Q) \in eSim-cs Rel \text{Cal} \) |
right: \( [P \rightarrow \text{Cal} \times P'; (Q, P) \in R] \rightarrow (P', Q) \in eSim-cs Rel \text{Cal} \) |
trans: \( [(P, Q) \in eSim-cs Rel \text{Cal}; (Q, R) \in eSim-cs Rel \text{Cal}] \Rightarrow (P, R) \in eSim-cs Rel \text{Cal} \)

lemma weak-reduction-correspondence-simulation-impl-coupled-simulation:
fixes Rel : (proc \( \times \) proc set)
and \( \text{Cal} : \text{proc processCalculus} \)
assumes \( \text{corrSim} : \text{weak-reduction-correspondence-simulation Rel Cal} \)
shows weak-reduction-coupled-simulation (eSim-cs Rel Cal) \( \text{Cal} \)
and \( \forall P Q. (P, Q) \in R \rightarrow (P, Q) \in eSim-cs Rel \text{Cal} \land (Q, P) \in eSim-cs Rel \text{Cal} \)
proof -
show weak-reduction-coupled-simulation (eSim-cs Rel Cal) \( \text{Cal} \)
proof (rule allI, rule allI, rule allI, rule impI, erule conjE)
fix $P \ Q \ P'$
assume $(P, Q) \in \text{cSim-cs Rel Cal}$ and $P \rightarrow \text{Cal*} \ P'$
thus $(\exists \ Q' \ Q \rightarrow \text{Cal*} \ Q' \land (P', Q') \in \text{cSim-cs Rel Cal})$
\land $(\exists Q' \ Q \rightarrow \text{Cal*} \ Q' \land (Q', P) \in \text{cSim-cs Rel Cal})$
proof (induct arbitrary: $P'$)
\begin{itemize}
\item case (left $Q \ Q' \ P$)
\begin{itemize}
\item assume $(P, Q') \in \text{Rel}$ and $P \rightarrow \text{Cal*} \ P'$
\end{itemize}
\begin{itemize}
\item with $\text{corrSim}$ obtain $Q''$ where $A1: Q' \rightarrow \text{Cal*} \ Q''$ and $A2: (P', Q'') \in \text{Rel}$
\end{itemize}
\begin{itemize}
\item by blast
\end{itemize}
\item assume $A3: Q \rightarrow \text{Cal*} \ Q'$
\item from this $A1$ have $A4: Q \rightarrow \text{Cal*} \ Q''$
\item by $(\text{rule steps-add}[\text{where } P=Q \land Q=Q' \land R=R'])$
\item have $Q'' \rightarrow \text{Cal*} \ Q''$
\item by $(\text{rule steps-refl})$
\item with $A2$ have $A5: (Q'', P') \in \text{cSim-cs Rel Cal}$
\item by $(\text{simp add: cSim-cs.right})$
\item from $A1 \ A2$ have $(P', Q') \in \text{cSim-cs Rel Cal}$
\item by $(\text{rule cSim-cs.left})$
\item with $A4 \ A5 \ A3$ show ?case
\item by blast
\end{itemize}
\item next
\begin{itemize}
\item case (right $P \ P' \ Q \ P''$)
\item assume $P \rightarrow \text{Cal*} \ P'$ and $P' \rightarrow \text{Cal*} \ P''$
\item hence $B1: P \rightarrow \text{Cal*} \ P''$
\item by $(\text{rule steps-add}[\text{where } P=P \land Q=P' \land R=P''])$
\item assume $B2: (Q, P) \in \text{Rel}$
\item with $\text{corrSim} \ B1$ obtain $Q'' \ P''$ where $B3: Q \rightarrow \text{Cal*} \ Q''$ and $B4: P'' \rightarrow \text{Cal*} \ P''$
\item and $B5: (Q'', P'') \in \text{Rel}$
\item by blast
\item from $B3 \ B5$ have $B6: (Q'', P'') \in \text{cSim-cs Rel Cal}$
\item by $(\text{rule cSim-cs.left})$
\item have $B7: Q \rightarrow \text{Cal*} \ Q$
\item by $(\text{rule steps-refl})$
\item from $B1 \ B2$ have $(P'', Q) \in \text{cSim-cs Rel Cal}$
\item by $(\text{rule cSim-cs.right})$
\item with $B3 \ B6 \ B7$ show ?case
\item by blast
\end{itemize}
\item next
\begin{itemize}
\item case (trans $P \ Q \ R \ P'$)
\item assume $P \rightarrow \text{Cal*} \ P'$
\item and $(\exists P', P \rightarrow \text{Cal*} \ P' \Rightarrow (\exists Q', Q \rightarrow \text{Cal*} \ Q' \land (P', Q') \in \text{cSim-cs Rel Cal})$
\item \land $(\exists Q', Q \rightarrow \text{Cal*} \ Q' \land (Q', P') \in \text{cSim-cs Rel Cal})$
\item from this obtain $Q1 \ Q2$ where $C1: Q \rightarrow \text{Cal*} \ Q1$ and $C2: (Q1, P') \in \text{cSim-cs Rel Cal}$
\item and $C3: Q \rightarrow \text{Cal*} \ Q2$ and $C4: (P', Q2) \in \text{cSim-cs Rel Cal}$
\item by blast
\item assume $C5: \ (\exists Q'. \ Q \rightarrow \text{Cal*} \ Q' \Rightarrow (\exists R'. \ R \rightarrow \text{Cal*} \ R' \land (Q', R') \in \text{cSim-cs Rel Cal})$
\item \land $(\exists R'. \ R \rightarrow \text{Cal*} \ R' \land (R', Q') \in \text{cSim-cs Rel Cal})$
\item with $C1$ obtain $R1$ where $C6: R \rightarrow \text{Cal*} \ R1$ and $C7: (R1, Q1) \in \text{cSim-cs Rel Cal}$
\item by blast
\item from $C7 \ C2$ have $C8: (R1, P') \in \text{cSim-cs Rel Cal}$
\item by $(\text{rule cSim-cs.trans})$
\item from $C3 \ C5$ obtain $R2$ where $C9: R \rightarrow \text{Cal*} \ R2$ and $C10: (Q2, R2) \in \text{cSim-cs Rel Cal}$
\item by blast
\item from $C4 \ C10$ have $(P', R2) \in \text{cSim-cs Rel Cal}$
\item by $(\text{rule cSim-cs.trans})$
\item with $C6 \ C8 \ C9$ show ?case
\item by blast
\end{itemize}
\begin{itemize}
\item qed
\end{itemize}
\begin{itemize}
\item qed
\end{itemize}
\begin{itemize}
\item next
\item show $(\forall \ P \ Q. \ (P, Q) \in \text{Rel} \rightarrow (P, Q) \in \text{cSim-cs Rel Cal} \land (Q, P) \in \text{cSim-cs Rel Cal})$
\end{itemize}
proof clarify
fix P Q
have Q |→ Cal* Q
by (rule steps-refl)
moreover assume (P, Q) ∈ Rel
ultimately show (P, Q) ∈ cSim-cs Rel Cal ∧ (Q, P) ∈ cSim-cs Rel Cal
by (simp add: cSim-cs.left cSim-cs.right)
qed
qed

lemma weak-barbed-correspondence-simulation-impl-coupled-simulation:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes corrSim: weak-barbed-correspondence-simulation Rel CWB
shows weak-barbed-coupled-simulation (cSim-cs Rel (Calculus CWB)) CWB
and ∀ P Q. (P, Q) ∈ Rel |→ (P, Q) ∈ cSim-cs Rel (Calculus CWB)
∧ (Q, P) ∈ cSim-cs Rel (Calculus CWB)
proof
from corrSim
show weak-reduction-coupled-simulation (cSim-cs Rel (Calculus CWB)) (Calculus CWB)
using weak-reduction-correspondence-simulation-impl-coupled-simulation(1)[where Rel=Rel
and Cal=Calculus CWB]
by blast
next
show rel-weakly-preserves-barbs (cSim-cs Rel (Calculus CWB)) CWB
proof clarify
fix P Q a P'
assume (P, Q) ∈ cSim-cs Rel (Calculus CWB) and P |→(Calculus CWB)* P' and P'↓<CWB>a
thus Q↓<CWB>a
proof (induct arbitrary: P')
case (left Q Q' P P')
assume (P, Q') ∈ Rel and P |→(Calculus CWB)* P' and P'↓<CWB>a
with corrSim obtain Q'' where A1: Q' |→(Calculus CWB)* Q'' and A2: Q''↓<CWB>a
by blast
assume Q |→(Calculus CWB)* Q'
from this A1 have Q |→(Calculus CWB)* Q''
by (rule steps-add)
with A2 show Q↓<CWB>a
by blast
next
case (right P P' Q P''')
assume (Q, P) ∈ Rel
moreover assume P |→(Calculus CWB)* P' and P' |→(Calculus CWB)* P''
hence P |→(Calculus CWB)* P''
by (rule steps-add)
moreover assume P''↓<CWB>a
ultimately show Q↓<CWB>a
using corrSim
by blast
next
case (trans P Q R P')
assume ∧ P'. P |→(Calculus CWB)* P' |→<CWB>a Q |→<CWB>a
and P |→(Calculus CWB)* P' and P'↓<CWB>a
and ∧ Q'. Q |→(Calculus CWB)* Q' |→<CWB>a R |→<CWB>a
thus R↓<CWB>a
by blast
qed
qed
qed
The reflexive and/or transitive closure of a weak correspondence simulation is a weak correspondence simulation.

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

- **fixes** \( Cal = \text{'proc process Calculus} \)
- **and** \( P Q R = \text{'proc} \)
- **and** \( Rel = (\text{'proc } \times \text{'proc}) \cdot \text{set} \)

**assumes**

1. \( A1: \forall Q'. Q \leftrightarrow (\exists P'' Q''). P \leftrightarrow \text{Cal} P'' \land Q' \leftrightarrow \text{Cal} Q'' \land (P'', Q'') \in \text{Rel} \)

2. \( A2: \forall R'. R \leftrightarrow \text{Cal} R' \leftrightarrow (\exists Q'' R''). Q \leftrightarrow \text{Cal} Q'' \land R' \leftrightarrow \text{Cal} R'' \land (Q'', R'') \in \text{Rel} \)

3. \( A3: \) weak-reduction-correspondence-simulation-Rel Cal

4. \( A4: \) trans Rel

**shows**

\( \forall R'. R \leftrightarrow \text{Cal} R' \leftrightarrow (\exists P'' R''). P \leftrightarrow \text{Cal} P'' \land R' \leftrightarrow \text{Cal} R'' \land (P'', R'') \in \text{Rel} \)

**proof** clarify

- **fix** \( R' \)
- **assume** \( R \leftrightarrow \text{Cal} R' \)
- **with** \( A2 \) obtain \( Q'' R'' \) where \( A5: Q \leftrightarrow (\exists P'' Q''). P \leftrightarrow (\text{Cal} P'' \land Q' \leftrightarrow (\text{Cal} Q'' \land (P'', Q'') \in \text{Rel}) \)

- **and** \( A6: R' \leftrightarrow \text{Cal} R'' \)

- **and** \( A7: (Q'', R'') \in \text{Rel} \)

- **by blast**

- **from** \( A1 \) obtain \( P''' Q''' \) where \( A8: P \leftrightarrow (\text{Cal} P''') \) and \( A9: Q'' \leftrightarrow (\text{Cal} Q''') \)

- **and** \( A10: (P''', Q''') \in \text{Rel} \)

- **by blast**

- **from** \( A3 \) \( A7 \) \( A9 \) obtain \( R''' \) where \( A11: R'' \leftrightarrow (\text{Cal} R''') \) and \( A12: (Q''', R''') \in \text{Rel} \)

- **by blast**

- **from** \( A6 \) \( A11 \) have \( A13: R' \leftrightarrow (\text{Cal} R''') \)

- **by** (rule steps-add [where \( P = R' \) and \( Q = R'' \) and \( R = R''' \)])

- **from** \( A4 \) \( A10 \) \( A12 \) have \( (P''', R''') \in \text{Rel} \)

- **unfolding trans-def**

- **by blast**

- **with** \( A8 \) \( A13 \) show \( \exists P'' R''. P \leftrightarrow \text{Cal} P''' \land R' \leftrightarrow \text{Cal} R'' \land (P'', R'') \in \text{Rel} \)

- **by blast**

**qed**

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

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

- **fixes** \( Rel = (\text{'proc } \times \text{'proc}) \cdot \text{set} \)
- **and** \( Cal = \text{proc process Calculus} \)

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

**shows** weak-reduction-correspondence-simulation-Rel+ Cal

- **and** weak-reduction-correspondence-simulation-Rel- Cal

- **and** weak-reduction-correspondence-simulation-(Rel±) Cal

**proof** —

**show** \( A: \) weak-reduction-correspondence-simulation-Rel= Cal

**proof**

- **from** \( \text{corrSim} \) show weak-reduction-simulation-Rel= Cal

- **using** weak-reduction-simulation-and-closures [where \( Rel = \text{Rel} = \text{Rel} \) and \( Cal = Cal \)]

**by blast**

**next**

**show** 

\( \forall P Q Q'. (P, Q) \in \text{Rel} = Q \leftrightarrow (\exists P'' Q''). P \leftrightarrow (\text{Cal} P'' \land Q' \leftrightarrow (\text{Cal} Q'' \land (P'', Q'') \in \text{Rel} =) \)

**proof** clarify

- **fix** \( P Q Q' \)

- **assume** \( (P, Q) \in \text{Rel} = \) and \( A1: Q \leftrightarrow \text{Cal} Q' \)

- **moreover have** \( P = Q \Rightarrow \exists P'' Q''. P \leftrightarrow \text{Cal} P'' \land Q' \leftrightarrow \text{Cal} Q'' \land (P'', Q'') \in \text{Rel} = \)

**proof** —
proof

assume \( P = Q \)

moreover have \( Q' \mapsto \text{Cal}^* Q' \)

by (rule steps-refl)

ultimately show \( \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

using A1 refl

by blast

qed

moreover

have \( (P, Q) \in \text{Rel} \implies \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

proof

- assume \( (P, Q) \in \text{Rel} \)

with \text{corrSim} A1 obtain \( P'' Q'' \) where \( P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

by blast

thus \( \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

by auto

qed

ultimately show \( \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

by auto

qed

have \( B : \land \text{Rel}. \text{weak-reduction-correspondence-simulation} \text{Rel} \text{Cal} \implies \text{weak-reduction-correspondence-simulation} (\text{Rel}^+) \text{Cal} \)

proof

fix \text{Rel}

assume \text{weak-reduction-correspondence-simulation} \text{Rel} \text{Cal}

thus \text{weak-reduction-simulation} (\text{Rel}^+) \text{Cal}

using \text{weak-reduction-simulation-and-closures}(2)[\text{where Rel}=\text{Rel} \text{and Cal}=\text{Cal}]

by blast

next

fix \text{Rel}

assume B1: \text{weak-reduction-correspondence-simulation} \text{Rel} \text{Cal}

show \( \forall P Q Q'. (P, Q) \in \text{Rel}^+ \land Q \mapsto \text{Cal}^* Q' \mapsto (\exists P'' Q''). P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

proof clarify

fix \( P \ Q \ Q' \)

assume \( (P, Q) \in \text{Rel}^+ \) \text{and} \( Q \mapsto \text{Cal}^* Q' \)

thus \( \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

proof (induct arbitrary: \( Q' \))

fix \( Q \ Q' \)

assume \( (P, Q) \in \text{Rel} \) \text{and} \( Q \mapsto \text{Cal}^* Q' \)

with B1 obtain \( P'' Q'' \) where B2: \( P \mapsto \text{Cal}^* P'' \) \text{and} B3: \( Q' \mapsto \text{Cal}^* Q'' \)

and B4: \( (P'', Q'') \in \text{Rel}^+ \)

by blast

from B4 have \( (P'', Q'') \in \text{Rel}^+ \)

by simp

with B2 B3 show \( \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

by blast

next

case (step \( Q \ R \ R' \))

assume \( \land Q', \ Q \mapsto \text{Cal}^* Q' \)

\( \implies \exists P'' Q''. P \mapsto \text{Cal}^* P'' \land Q' \mapsto \text{Cal}^* Q'' \land (P'', Q'') \in \text{Rel}^+ \)

moreover assume \( (Q, R) \in \text{Rel} \)

with B1

have \( \land R', \ R \mapsto \text{Cal}^* R' \implies \exists Q'' Q'R''. Q \mapsto \text{Cal}^* Q'' \land R' \mapsto \text{Cal}^* R'' \land (Q'', R'') \in \text{Rel}^+ \)

by blast

moreover from B1 have \text{weak-reduction-simulation} (\text{Rel}^+) \text{Cal}

using \text{weak-reduction-simulation-and-closures}(2)[\text{where Rel}=\text{Rel} \text{and Cal}=\text{Cal}]

by blast

moreover have \text{trans} \text{Rel}^+)

using \text{trans-trancl}[\text{of Rel}]
A weak reduction bisimulation is relation $R$ such that (1) if $(P, Q)$ in $R$ and $P$ evolves to some $P'$ then there exists some $Q'$ such that $Q$ evolves to $Q'$ and $(P', Q')$ in $R$, and (2) if $(P, Q)$ in $R$ and $Q$ evolves to some $Q'$ then there exists some $P'$ such that $P$ evolves to $P'$ and $(P', Q')$ in $R$.

**abbreviation** weak-reduction-bisimulation

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

where

weak-reduction-bisimulation $Rel \ Cal \equiv$

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

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

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

**abbreviation** weak-barbed-bisimulation

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

where

weak-barbed-bisimulation $Rel \ CWB \equiv$
A symmetric weak simulation is a weak bisimulation.

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

- **fixes** `Rel :: ('proc × 'proc) set`
- **and** `Cal :: 'proc processCalculus`
- **assumes** `sym Rel`
  - **and** `weak-reduction-simulation Rel Cal`
- **shows** `weak-reduction-bisimulation Rel Cal`
  - **using** `assms symD [of Rel]`
  - **by** `blast`

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

- **fixes** `Rel :: ('proc × 'proc) set`
- **and** `CWB :: ('proc, 'barbs) calculusWithBarbs`
- **assumes** `sym Rel`
  - **and** `weak-barbed-simulation Rel Cal`
- **shows** `weak-barbed-bisimulation Rel Cal`
  - **using** `assms symD [of Rel]`
  - **by** `blast`

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

**lemma** weak-reduction-simulations-impl-bisimulation:

- **fixes** `Rel :: ('proc × 'proc) set`
- **and** `Cal :: 'proc processCalculus`
- **assumes** `sim :: weak-reduction-simulation Rel Cal`
  - **and** `simInv :: weak-reduction-simulation (Rel⁻¹) Cal`
- **shows** `weak-reduction-bisimulation Rel Cal`
  - **proof** `auto`
  - **fix** `P Q P'`
    - **assume** `(P, Q) ∈ Rel and P ⟷ Cal* P'`
    - **with** `sim show ∃ Q'. Q ⟷ Cal* Q' ∧ (P', Q') ∈ Rel`
      - **by** `simp`
  - **next**
    - **fix** `P Q Q'`
    - **assume** `(P, Q) ∈ Rel`
    - **hence** `(Q, P) ∈ Rel⁻¹`
      - **by** `simp`
    - **moreover assume** `Q ⟷ Cal* Q'`
      - **ultimately obtain** `P' where A1: P ⟷ Cal* P' and A2: (Q', P') ∈ Rel⁻¹`
        - **using** `simInv`
        - **by** `blast`
    - **from** `A2` **have** `(P', Q') ∈ Rel`
      - **by** `induct`
    - **with** `A1 show ∃ P'. P ⟷ Cal* P' ∧ (P', Q') ∈ Rel`
      - **by** `blast`
  - **qed**

**lemma** weak-reduction-bisimulations-impl-inverse-is-simulation:

- **fixes** `Rel :: ('proc × 'proc) set`
- **and** `Cal :: 'proc processCalculus`
- **assumes** `bisim :: weak-reduction-bisimulation Rel Cal`
- **shows** `weak-reduction-simulation (Rel⁻¹) Cal`
  - **proof** `clarify`
    - **fix** `P Q P'`
    - **assume** `(Q, P) ∈ Rel`
    - **moreover assume** `P ⟷ Cal* P'`
    - **ultimately obtain** `Q' where A1: Q ⟷ Cal* Q' and A2: (Q', P') ∈ Rel⁻¹`
      - **using** `bisim`
      - **by** `blast`
from $A^2$ have $(P', Q') \in \text{Rel}^{-1}$
by simp
with A1 show $\exists Q'. Q \mapsto_{\text{Cal}} Q' \land (P', Q') \in \text{Rel}^{-1}$
by blast
qed

lemma weak-reduction-simulations-iff-bisimulation:
fixes Rel :: ('proc' \times 'proc) set
and Cal :: 'proc processCalculus
shows (weak-reduction-simulation Rel Cal \land weak-reduction-simulation (\text{Rel}^{-1}) Cal) 
= weak-reduction-bisimulation Rel Cal
using weak-reduction-simulations-impl-bisimulation[where Rel=Rel and Cal=Cal]
weak-reduction-bisimulations-impl-inverse-is-simulation[where Rel=Rel and Cal=Cal]
by blast

lemma weak-barbed-simulations-iff-bisimulation:
fixes Rel :: ('proc' \times 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
shows (weak-barbed-simulation Rel CWB \land weak-barbed-simulation (\text{Rel}^{-1}) CWB) 
= weak-barbed-bisimulation Rel CWB
proof (rule iffI, erule conjE)
assume sim: weak-barbed-simulation Rel CWB
and rev: weak-barbed-simulation (\text{Rel}^{-1}) CWB
hence weak-reduction-bisimulation Rel (\text{Calculus CWB})
using weak-reduction-simulations-impl-bisimulation[where Rel=Rel and Cal=\text{Calculus CWB}]
by blast
moreover from sim have rel-weakly-preserves-barbs Rel CWB
by simp
moreover from rev have rel-weakly-reflects-barbs Rel CWB
by simp
ultimately show weak-barbed-bisimulation Rel CWB
by blast
next
assume bisim: weak-barbed-bisimulation Rel CWB
hence weak-barbed-simulation Rel CWB
by blast
moreover from bisim have weak-reduction-simulation (\text{Rel}^{-1}) (\text{Calculus CWB})
using weak-reduction-bisimulations-impl-inverse-is-simulation[where Rel=Rel]
by simp
moreover from bisim have rel-weakly-reflects-barbs Rel CWB
by blast
hence rel-weakly-preserves-barbs (\text{Rel}^{-1}) CWB
by simp
ultimately show weak-barbed-simulation Rel CWB \land weak-barbed-simulation (\text{Rel}^{-1}) CWB
by blast
qed

A weak bisimulation is a weak correspondence simulation.

lemma weak-reduction-bisimulation-is-correspondence-simulation:
fixes Rel :: ('proc' \times 'proc) set
and Cal :: 'proc processCalculus
assumes bisim: weak-reduction-bisimulation Rel Cal
shows weak-reduction-correspondence-simulation Rel Cal
proof
from bisim show weak-reduction-simulation Rel Cal
by blast
next
show $\forall P Q Q'. (P, Q) \in \text{Rel} \land Q \mapsto_{\text{Cal}} Q' \land (P', Q') \in \text{Rel}$
proof clarify
fix $P Q Q'$

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

lemma weak-barbed-bisimulation-is-correspondence-simulation:
fixes Rel :: ('proc × 'proc) set
and CWB :: ('proc, 'barbs) calculusWithBarbs
assumes bisim: weak-barbed-bisimulation Rel CWB
shows weak-barbed-correspondence-simulation Rel CWB
  using bisim weak-reduction-bisimulation-is-correspondence-simulation[where Rel=Rel
  and Cal=Calculus CWB]
by blast

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

lemma weak-reduction-bisimulation-and-closures:
fixes Rel :: ('proc × 'proc) set
and Cal :: 'proc processCalculus
assumes bisim: weak-reduction-bisimulation Rel Cal
shows weak-reduction-bisimulation (Rel*) Cal
  and weak-reduction-bisimulation (symcl Rel) Cal
  and weak-reduction-bisimulation (Rel+) Cal
  and weak-reduction-bisimulation (symcl (Rel*)) Cal
  and weak-reduction-bisimulation (symcl (Rel*)) Cal
  and weak-reduction-bisimulation (symcl (Rel*)) Cal
proof
  from bisim show A: weak-reduction-bisimulation (Rel*) Cal
    by (auto simp add: refl, blast+)
  have B: ∀Rel. weak-reduction-bisimulation Rel Cal
    ⇒ weak-reduction-bisimulation (symcl Rel) Cal
    by (auto simp add: symcl-def, blast+)
  from bisim B[where Rel=Rel] show weak-reduction-bisimulation (symcl Rel) Cal
    by blast
  have C: ∀Rel. weak-reduction-bisimulation Rel Cal
    ⇒ weak-reduction-bisimulation (Rel+) Cal
proof
  fix Rel
  assume weak-reduction-bisimulation Rel Cal
  thus weak-reduction-simulation (Rel+) Cal
    using weak-reduction-simulation-and-closures(2)[where Rel=Rel and Cal=Cal]
    by blast
next
  fix Rel
  assume C1: weak-reduction-bisimulation Rel Cal
  show ∀P Q Q'. (P, Q) ∈ Rel* ∧ Q → Cal* Q' → (∃P'. P → Cal* P' ∧ (P', Q') ∈ Rel*)
proof clarify
  fix P Q Q'
  assume (P, Q) ∈ Rel* and Q → Cal* Q'
  thus ∃P'. P → Cal* P' ∧ (P', Q') ∈ Rel*
  proof (induct arbitrary: Q')
    fix Q Q'
    assume (P, Q) ∈ Rel* and Q → Cal* Q'
    with C1 obtain P' where P → Cal* P' and (P', Q') ∈ Rel
    by blast
    thus ∃P'. P → Cal* P' ∧ (P', Q') ∈ Rel*
lemma weak-barbed-bisimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: weak-barbed-bisimulation Rel CWB
  shows weak-barbed-bisimulation (Rel⁻) 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 ((symcl (Rel⁻))⁺) CWB
  proof
    from bisim show weak-barbed-bisimulation (Rel⁻) CWB
      using weak-reduction-bisimulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
      weak-resection-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]
      by fast
    next
    from bisim show weak-barbed-bisimulation (symcl Rel) CWB
      using weak-reduction-bisimulation-and-closures(2)[where Rel=Rel and Cal=Calculus CWB]
      weak-resection-of-barbs-and-closures(2)[where Rel=Rel and CWB=CWB]
      by blast
    next
    from bisim show weak-barbed-bisimulation (Rel⁺) CWB
      using weak-reduction-bisimulation-and-closures(3)[where Rel=Rel and Cal=Calculus CWB]
      weak-resection-of-barbs-and-closures(3)[where Rel=Rel and CWB=CWB]
      by blast
    next
    from bisim show weak-barbed-bisimulation (symcl (Rel⁻)) CWB
      using weak-reduction-bisimulation-and-closures(4)[where Rel=Rel and Cal=Calculus CWB]
      weak-resection-of-barbs-and-closures(4)[where Rel=Rel and CWB=CWB]
      by blast
    next
    from bisim show weak-barbed-bisimulation (Rel⁺) CWB
      using weak-reduction-bisimulation-and-closures(5)[where Rel=Rel and Cal=Calculus CWB]
      by blast
  qed
A strong reduction bisimulation is relation \( R \) such that (1) if \((P, Q) \in R\) and \( P' \) is a derivative of \( P \) then there exists some \( Q' \) such that \( Q' \) is a derivative of \( Q \) and \((P', Q') \in R\), and (2) if \((P, Q) \in R\) and \( Q' \) is a derivative of \( Q \) then there exists some \( P' \) such that \( P' \) is a derivative of \( P \) and \((P', Q') \in R\).

Abbreviation
\[
\text{strong-reduction-bisimulation} :: (\text{'proc} \times \text{'proc}) \rightarrow \text{processCalculus} \Rightarrow \text{bool}
\]
where
\[
\text{strong-reduction-bisimulation} \ Rel \ Cal \equiv \\
(\forall P Q P'. (P, Q) \in Rel \land P \rightarrow Cal P' \rightarrow (\exists Q'. Q \rightarrow Cal Q' \land (P', Q') \in Rel))
\land (\forall P Q Q'. (P, Q) \in Rel \land Q \rightarrow Cal Q' \rightarrow (\exists P'. P \rightarrow Cal P' \land (P', Q') \in Rel))
\]

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

Abbreviation
\[
\text{strong-barbed-bisimulation} :: (\text{'proc} \times \text{'proc}) \rightarrow (\text{'proc} \times \text{'barbs}) \rightarrow \text{calculusWithBarbs} \Rightarrow \text{bool}
\]
where
\[
\text{strong-barbed-bisimulation} \ Rel \ CWB \equiv \\
\text{strong-reduction-bisimulation} \ Rel \ (\text{Calculus CWB}) \land \text{rel-respects-barbs} \ Rel \ CWB
\]

A symmetric strong simulation is a strong bisimulation.

Lemma
\[
\text{symm-strong-reduction-simulation-is-bisimulation}:
\]
fixes \( Rel :: (\text{'proc} \times \text{'proc}) \rightarrow \text{processCalculus} \Rightarrow \text{bool} \)
and \( Cal :: \text{'proc} \Rightarrow \text{processCalculus} \Rightarrow \text{bool} \)
assumes \( \text{sym} \ Rel \)
and \( \text{strong-reduction-simulation} \ Rel \ Cal \)
shows \( \text{strong-reduction-bisimulation} \ Rel \ Cal \)
using \( \text{assms} \ \text{symD}[\text{of} \ Rel] \)
by blast

Lemma
\[
\text{symm-strong-barbed-simulation-is-bisimulation}:
\]
fixes \( Rel :: (\text{'proc} \times \text{'proc}) \rightarrow \text{processCalculus} \Rightarrow \text{bool} \)
and \( CWB :: (\text{'proc} \times \text{'barbs}) \rightarrow \text{calculusWithBarbs} \Rightarrow \text{bool} \)
assumes \( \text{sym} \ Rel \)
and \( \text{strong-barbed-simulation} \ Rel \ CWB \)
shows \( \text{strong-barbed-bisimulation} \ Rel \ CWB \)
using \( \text{assms} \ \text{symD}[\text{of} \ Rel] \)
by blast

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

Lemma
\[
\text{strong-reduction-simulations-impl-bisimulation}:
\]
fixes \( Rel :: (\text{'proc} \times \text{'proc}) \rightarrow \text{processCalculus} \Rightarrow \text{bool} \)
and \( Cal :: \text{'proc} \Rightarrow \text{processCalculus} \Rightarrow \text{bool} \)
assumes \( \text{sim} :: \text{strong-reduction-simulation} \ Rel \ Cal \)
and \( \text{simInv} :: \text{strong-reduction-simulation} \ (\text{Rel}^{-1}) \ Cal \)
shows \( \text{strong-reduction-bisimulation} \ Rel \ Cal \)
proof auto
fix \( P Q P' \)
assume \( (P, Q) \in Rel \) and \( P \rightarrow Cal P' \)
with \( \text{sim} \) show \( \exists Q'. Q \rightarrow Cal Q' \land (P', Q') \in Rel \)
by simp
next
fix $P \, Q \, Q'$
assume $(P, Q) \in \text{Rel}
hence $(Q, P) \in \text{Rel}^{-1}$
   by simp
moreover assume $Q \mapsto \text{Cal} \, Q'$
ultimately obtain $P'$ where $A1: P \mapsto \text{Cal} \, P'$ and $A2: (Q', P') \in \text{Rel}^{-1}$
   using simInv
by blast
from $A2$ have $(P', Q') \in \text{Rel}$
   by induct
with $A1$ show $\exists \, P'. \, P \mapsto \text{Cal} \, P'$ and $(P', Q') \in \text{Rel}$
   by blast
qed

lemma strong-reduction-bisimulations-impl-inverse-is-simulation:
fixes $\text{Rel} : (\text{'proc} \times \text{'proc}) \text{set}$
and $\text{Cal} : \text{'proc processCalculus}$
assumes $\text{bisim: strong-reduction-bisimulation} \, \text{Rel} \, \text{Cal}$
shows $\text{strong-reduction-simulation} \, (\text{Rel}^{-1}) \, \text{Cal}$
proof (clarify)
fix $P \, Q \, P'$
assume $(Q, P) \in \text{Rel}$
moreover assume $P \mapsto \text{Cal} \, P'$
ultimately obtain $Q'$ where $A1: Q \mapsto \text{Cal} \, Q'$ and $A2: (Q', P') \in \text{Rel}$
   using simInv
by blast
from $A2$ have $(P', Q') \in \text{Rel}^{-1}$
   by simp
with $A1$ show $\exists \, Q'. \, Q \mapsto \text{Cal} \, Q'$ and $(P', Q') \in \text{Rel}^{-1}$
   by blast
qed

lemma strong-reduction-simulations-iff-bisimulation:
fixes $\text{Rel} : (\text{'proc} \times \text{'proc}) \text{set}$
and $\text{Cal} : \text{'proc processCalculus}$
shows $(\text{strong-reduction-simulation} \, \text{Rel} \, \text{Cal} \lor \text{strong-reduction-simulation} \, (\text{Rel}^{-1}) \, \text{Cal})$
   $\Rightarrow$ $\text{strong-reduction-bisimulation} \, \text{Rel} \, \text{Cal}$
using strong-reduction-simulations-impl-bisimulation$[\text{where} \text{Rel}=\text{Rel} \land \text{Cal}=\text{Cal}]$
strong-reduction-bisimulations-impl-inverse-is-simulation$[\text{where} \text{Rel}=\text{Rel}]$
by blast

lemma strong-barbed-simulations-iff-bisimulation:
fixes $\text{Rel} : (\text{'proc} \times \text{'proc}) \text{set}$
and $\text{CWB} : (\text{'proc}, \text{'barbs}) \text{calculusWithBarbs}$
shows $(\text{strong-barbed-simulation} \, \text{Rel} \, \text{CWB} \lor \text{strong-barbed-simulation} \, (\text{Rel}^{-1}) \, \text{CWB})$
   $\Rightarrow$ $\text{strong-barbed-bisimulation} \, \text{Rel} \, \text{CWB}$
proof (rule iffI, erule conjE)
assume sim: $\text{strong-barbed-simulation} \, \text{Rel} \, \text{CWB}$
and rev: $\text{strong-barbed-simulation} \, (\text{Rel}^{-1}) \, \text{CWB}$
hence $\text{strong-reduction-bisimulation} \, \text{Rel} \, (\text{Calculus} \, \text{CWB})$
   using strong-reduction-simulations-impl-bisimulation$[\text{where} \text{Rel}=\text{Rel} \land \text{Cal}=\text{Calculus} \, \text{CWB}]$
by blast
moreover from sim have rel-preserves-barbs Rel $\, \text{CWB}$
   by simp
moreover from rev have rel-reflects-barbs Rel $\, \text{CWB}$
   by simp
ultimately show $\text{strong-barbed-bisimulation} \, \text{Rel} \, \text{CWB}$
   by blast
next
assume bisim: $\text{strong-barbed-bisimulation} \, \text{Rel} \, \text{CWB}$
hence $\text{strong-barbed-simulation} \, \text{Rel} \, \text{CWB}$
by blast
moreover from bisim have strong-reduction-simulation \((Rel^{-1})\) \((Calculus\ CWB)\)
  using strong-reduction-bisimulations-impl-inverse-is-simulation\[\text{where } Rel = Rel\]
by simp
moreover from bisim have rel-reflects-barbs \(Rel\ CWB\)
by blast
hence rel-preserves-barbs \((Rel^{-1})\) \(CWB\)
by simp
ultimately
show strong-barbed-simulation \(Rel\ CWB\) \(\land\) strong-barbed-simulation \((Rel^{-1})\) \(CWB\)
by blast
qed

A strong bisimulation is a weak bisimulation.

**Lemma strong-impl-weak-reduction-bisimulation:**
fixes \(Rel :: ('proc \times 'proc)\) set
  and \(Cal :: 'proc\ process\ calculus\)
assumes bisim: strong-reduction-bisimulation \(Rel\ Cal\)
shows weak-reduction-bisimulation \(Rel\ Cal\)
proof
  from bisim show weak-reduction-simulation \(Rel\ Cal\)
  using strong-impl-weak-reduction-simulation \[\text{where } Rel = Rel\]
  by blast
next
  show \(\forall P Q Q'.\ (P, Q) \in Rel \land Q \mapsto Cal^* Q' \mapsto (\exists P'.\ P \mapsto Cal^* P' \land (P', Q') \in Rel)\)
  proof clarify
    fix \(P Q Q'\)
    assume A1: \((P, Q) \in Rel\)
    assume Q \mapsto Cal^* Q'
    from this obtain \(n\) where \(Q \mapsto Cal^n Q'\)
    by (auto simp add: steps-def)
    thus \(\exists P'.\ P \mapsto Cal^* P' \land (P', Q') \in Rel\)
  proof (induct \(n\) arbitrary: \(Q'\))
    case 0
    assume Q \mapsto Cal^0 Q'
    hence \(Q = Q'\)
    moreover have \(P \mapsto Cal^* P\)
    by (rule steps-refl)
    ultimately show \(\exists P'.\ P \mapsto Cal^* P' \land (P', Q') \in Rel\)
      using A1
    by blast
next
  case \((Suc n Q')\)
  assume Q \mapsto Cal^Suc n Q''
  from this obtain \(Q'\) where A2: \(Q \mapsto Cal^n Q'\) \(\land\) A3: \(Q' \mapsto Cal Q''\)
  by auto
  assume \(\land Q'.\ Q \mapsto Cal^n Q' \mapsto P'.\ P \mapsto Cal^* P' \land (P', Q') \in Rel\)
  with A2 obtain \(P'\) where A4: \(P \mapsto Cal^* P'\) \(\land\) A5: \(P', Q') \in Rel
  by blast
  from bisim A5 A3 obtain \(P''\) where A6: \(P' \mapsto Cal P''\) \(\land\) A7: \(P'', Q'') \in Rel
  by blast
  from A4 A6 have \(P \mapsto Cal^* P''\)
    using steps-add\[\text{where } P = P\] \(\land\) A3 \(\land\) A5 \(\land\) A7 \(\land\) A6 \(\land\) A5 \(\land\) A7
  by (simp add: steps-to-steps)
  with A7 show \(\exists P'.\ P \mapsto Cal^* P' \land (P', Q'') \in Rel\)
  by blast
qed
qed

qed
lemma strong-barbed-bisimulation-impl-weak-respection-of-barbs:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: strong-barbed-bisimulation Rel CWB
  shows rel-weakly-respects-barbs Rel CWB
proof
  from bisim show rel-weakly-preserves-barbs Rel CWB
  using strong-barbed-simulation-impl-weak-preservation-of-barbs
  [where Rel=Rel and CWB=CWB]
  by blast
next
  show rel-weakly-reflects-barbs Rel CWB
  proof
    clarify
    fix P Q a Q'
    assume (P, Q) ∈ Rel and Q ↠→(Calculus CWB)* Q'
    with bisim obtain P' where A1: P ↠→(Calculus CWB)* P' and
    A2: (P', Q') ∈ Rel
    using strong-impl-weak-reduction-bisimulation
    [where Rel=Rel and Cal=Calculus CWB]
    by blast
    assume Q'↓<CWB>a
    with bisim A2 have P'↓<CWB>a
    by blast
    with A1 show P⇓<CWB>a
    by blast
  qed
qed

lemma strong-impl-weak-barbed-bisimulation:
  fixes Rel :: ('proc × 'proc) set
  and CWB :: ('proc, 'barbs) calculusWithBarbs
  assumes bisim: strong-barbed-bisimulation Rel CWB
  shows weak-barbed-bisimulation Rel CWB
  using bisim
  strong-impl-weak-reduction-bisimulation
  [where Rel=Rel and Cal=Calculus CWB]
  strong-barbed-bisimulation-impl-weak-respection-of-barbs
  [where Rel=Rel and CWB=CWB]
  by blast

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

lemma strong-reduction-bisimulation-and-closures:
  fixes Rel :: ('proc × 'proc) set
  and Cal :: 'proc processCalculus
  assumes bisim: strong-reduction-bisimulation Rel Cal
  shows strong-reduction-bisimulation (Rel=) Cal
  and strong-reduction-bisimulation (symcl Rel) Cal
  and strong-reduction-bisimulation (Rel+) Cal
  and strong-reduction-bisimulation (symcl (Rel=)) Cal
  and strong-reduction-bisimulation (Rel*) Cal
  and strong-reduction-bisimulation ((symcl (Rel=))+) Cal
proof
  from bisim show A: strong-reduction-bisimulation (Rel=) Cal
  by (auto simp add: refl, blast+)
  have B: ∀Rel. strong-reduction-bisimulation Rel Cal
  ⇒ strong-reduction-bisimulation (symcl Rel) Cal
  by (auto simp add: symcl-def, blast+)
  from bisim B[where Rel=Rel] show strong-reduction-bisimulation (symcl Rel) Cal
  by blast
  have C: ∀Rel. strong-reduction-bisimulation Rel Cal
  ⇒ strong-reduction-bisimulation (Rel+) Cal
  proof
    fix Rel
    assume strong-reduction-bisimulation Rel Cal
    thus strong-reduction-simulation (Rel+) Cal
using strong-reduction-simulation-and-closures(2) [where Rel=Rel and Cal=Cal]

by blast

next
fix Rel
assume C1: strong-reduction-bisimulation Rel Cal
show \( \forall P \ Q \ Q'. (P, Q) \in Rel^{+} \land Q \longrightarrow Cal Q' \rightarrow (\exists P'. P \longrightarrow Cal P' \land (P', Q') \in Rel^{+}) \)
proof clarify
fix P Q Q'
assume \((P, Q) \in Rel^{+} \land Q \longrightarrow Cal Q'\)
thus \( \exists P'. P \longrightarrow Cal P' \land (P', Q') \in Rel^{+} \)
proof (induct arbitrary: \(Q')\)
fix Q Q'
assume \((P, Q) \in Rel \land Q \longrightarrow Cal Q'\)
with C1 obtain \(P' \) where \(P \longrightarrow Cal P' \land (P', Q') \in Rel\)
by blast
thus \( \exists P'. P \longrightarrow Cal P' \land (P', Q') \in Rel^{+} \)
by auto
next
case (step \(Q R R'\))
assume \((Q, R) \in Rel \land R \longrightarrow Cal R'\)
with C1 obtain \(Q' \) where C2: \(Q \longrightarrow Cal Q'\) and C3: \((Q', R') \in Rel^{+}\)
by blast
assume \( \land Q', Q \longrightarrow Cal Q' \rightarrow (\exists P'. P \longrightarrow Cal P' \land (P', Q') \in Rel^{+} \)
with C2 obtain \(P' \) where C4: \(P \longrightarrow Cal P' \land (P', Q') \in Rel^{+}\)
by blast
from C5 C3 have \((P', R') \in Rel^{+}\)
by simp
with C4 show \( \exists P'. P \longrightarrow Cal P' \land (P', R') \in Rel^{+} \)
by blast
qed

from bisim C[where Rel=Rel] show strong-reduction-bisimulation \((Rel^{+})\) Cal
by blast

from A B[where Rel=Rel’]
show strong-reduction-bisimulation \((symcl (Rel’))\) Cal
by blast

from A C[where Rel=Rel’]
show strong-reduction-bisimulation \((Rel^{+})\) Cal
using trancl-refl[of Rel]
by auto

from A B[where Rel=Rel’] C[where Rel=symcl (Rel’)]
show strong-reduction-bisimulation \((symcl (Rel’))^{+}\) Cal
by blast

qed

lemma strong-barbed-bisimulation-and-closures:
fixes Rel :: \(' 'proc \times ' 'proc\) set
and CWB :: \(' 'proc, ' 'barbs\) calculusWithBarbs
assumes bisim: strong-barbed-bisimulation Rel CWB
shows strong-barbed-bisimulation \((Rel’)\) CWB
and strong-barbed-bisimulation \((symcl Rel\) CWB
and strong-barbed-bisimulation \((Rel^{+})\) CWB
and strong-barbed-bisimulation \((symcl (Rel’))\) CWB
and strong-barbed-bisimulation \((Rel^{+})\) CWB
and strong-barbed-bisimulation \((symcl (Rel’))^{+}\) CWB
proof –
from bisim show strong-barbed-bisimulation \((Rel’)\) CWB
using strong-reduction-bisimulation-and-closures(1)[where Rel=Rel and Cal=Calculus CWB]
respection-of-barbs-and-closures(1)[where Rel=Rel and CWB=CWB]

39
Applying the steps closure twice does not change the relation.

**Abbreviation** stepsClosureInfix ::

\[
\text{P R} \Rightarrow\langle\text{stepsClosure Rel Cal},\text{Cal}\rangle \equiv (P, Q) \in \text{stepsClosure Rel Cal}
\]

Applying the steps closure twice does not change the relation.

**Lemma** steps-closure-of-steps-closure:

\[
\begin{align*}
\text{fixes} & \quad \text{Rel :: } (a \times a) \text{ set} \\
& \quad \text{and} \quad \text{Cal :: } a \text{ processCalculus} \\
\text{shows} & \quad \text{stepsClosure } (\text{stepsClosure Rel Cal}) \text{ Cal = stepsClosure Rel Cal} \\
\text{proof} & \quad \text{auto} \\
\end{align*}
\]

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.
next
case (steps P P')
assume P ↦→ Cal* P'
thus P' R↦→<Rel,Cal> P
  by (rule stepsClosure.steps)
next
case (trans P Q R)
assume P R↦→<Rel,Cal> Q and Q R↦→<Rel,Cal> R
thus P R↦→<Rel,Cal> R
  by (rule stepsClosure.trans)
qed

The steps closure is a preorder.

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

lemma refl-trans-closure-of-rel-impl-steps-closure:
  fixes Rel :: ('a × 'a) set
  and Cal :: 'a processCalculus
  and P Q :: 'a
  assumes (P, Q) ∈ Rel* 
  shows P R↦→<Rel,Cal> Q 
  using assms
proof induct
  show P R↦→<Rel,Cal> P 
    using stepsClosure-refl[of Rel Cal]
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume (Q, R) ∈ Rel and P R↦→<Rel,Cal> Q 
  thus P R↦→<Rel,Cal> R 
    using stepsClosure.rel[of Q R Rel Cal] stepsClosure.trans[of P Q Rel Cal R] 
    by blast
qed

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

lemma steps-closure-is-weak-reduction-simulation:
  fixes Rel :: ('a × 'a) set
  and Cal :: 'a processCalculus
  shows weak-reduction-simulation (stepsClosure Rel Cal) Cal
proof clarify
  fix P Q P'
  assume P R↦→<Rel,Cal> Q and P ↦→ Cal* P'

thus \( \exists Q'. Q \quad \text{cal} \quad Q' \land P' \quad \text{cal} \quad Q' \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**proof** (induct arbitrary: \( P' \))

**case** (rel \( P \ Q \))

**assume** \( P \quad \text{cal} \quad P' \)

**hence** \( P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad P \)

**by** (rule stepsClosure.steps)

**moreover assume** \( (P, Q) \in \text{Rel} \)

**hence** \( P \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q \)

**by** (simp add: stepsClosure.rel)

**ultimately have** \( P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**by** (rule stepsClosure.trans)

**thus** \( \exists Q'. Q \quad \text{cal} \quad Q' \land P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**using** steps-refl[where \( \text{Cal}=\text{Cal} \) and \( P=Q \)]

**by** blast

**next**

**case** (steps \( P \quad P' \quad P'' \))

**assume** \( P \quad \text{cal} \quad P' \) \quad \text{and} \quad \( P' \quad \text{cal} \quad P'' \)

**hence** \( P \quad \text{cal} \quad P'' \)

**by** (rule steps-add)

**moreover have** \( P'' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad P'' \)

**using** stepsClosure-refl[where \( \text{Rel}=\text{Rel} \) and \( \text{Cal}=\text{Cal} \)]

**unfolding** refl-on-def

**by** simp

**ultimately show** \( \exists Q'. P \quad \text{cal} \quad Q' \land P'' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**by** blast

**next**

**case** (trans \( P \quad Q \quad R \))

**assume** \( P \quad \text{cal} \quad P' \) \quad \text{and} \quad \( \bigwedge P'. P \quad \text{cal} \quad P' \quad \Rightarrow \quad \exists Q'. Q \quad \text{cal} \quad Q' \land P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**from this obtain** \( Q' \) \quad \text{where} \quad \text{A1:} \quad Q \quad \text{cal} \quad Q' \) \quad \text{and} \quad \text{A2:} \quad P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad Q' \)

**by** blast

**assume** \( \bigwedge Q'. Q \quad \text{cal} \quad Q' \quad \Rightarrow \quad \exists R'. R \quad \text{cal} \quad R' \land Q' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad R' \)

**with A1 obtain** \( R' \) \quad \text{where} \quad \text{A3:} \quad R \quad \text{cal} \quad R' \) \quad \text{and} \quad \text{A4:} \quad Q' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad R' \)

**by** blast

**from** \( A2 \ A4 \) **have** \( P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad R' \)

**by** (rule stepsClosure.trans)

**with** \( A3 \) **show** \( \exists R'. R \quad \text{cal} \quad R' \land P' \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad R' \)

**by** blast

**qed**

**qed**

If \( \text{Rel} \) is a weak simulation and its inverse is a weak contrasimulation, then the steps closure of \( \text{Rel} \) is a contrasimulation.

**lemma** inverse-contrasimulation-impl-reverse-pair-in-steps-closure:

**fixes** \( \text{Rel} :: \langle 'a \times 'a \rangle \) \quad \text{set}

**and** \( \text{Cal} :: \langle 'a \rangle \) \quad \text{processCalculus}

**and** \( P \quad Q :: \langle 'a \rangle \)

**assumes** \( \text{con:} \quad \text{weak-reduction-contrasimulation} \quad \langle \text{Rel}^{-1} \rangle \) \quad \text{Cal}

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

**shows** \( \text{Q} \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad P \)

**proof**

**–**

**from** \( \text{pair} \) **have** \( (Q, P) \in \text{Rel}^{-1} \)

**by** simp

**moreover have** \( Q \quad \text{cal} \quad Q \)

**by** (rule steps-refl)

**ultimately obtain** \( P' \) \quad \text{where} \quad \text{A1:} \quad P \quad \text{cal} \quad P' \) \quad \text{and} \quad \text{A2:} \quad (P', Q) \in \text{Rel}^{-1}

**using** \( \text{con} \)

**by** blast

**from** \( A2 \) **have** \( Q \quad \text{cal} \quad <\text{Rel}, \text{Cal}> \quad P' \)

**by** (simp add: stepsClosure.rel)
moreover from A1 have \( P' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P \)
by (rule stepsClosure.steps)
ultimately show \( Q \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P \)
by (rule stepsClosure.trans)
qed

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 clarify
fix P Q P'
assume \( P \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ Q \) and \( P \Longrightarrow \text{Cal} \ast P' \)
thus \( \exists Q'. \ Q \Longrightarrow \text{Cal} \ast Q' \land Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
proof (induct arbitrary: \( P' \))
case (rel P Q)
assume \( (P, Q) \in \text{Rel} \) and \( P \Longrightarrow \text{Cal} \ast P' \)
with sim obtain \( Q' \) where A1: \( Q \Longrightarrow \text{Cal} \ast Q' \) and A2: \( (P', Q') \in \text{Rel} \)
by blast
from A2 con have \( Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
using inverse-contrasimulation-impl-reverse-pair-in-steps-closure[where Rel=Rel]
by blast
with A1 show \( \exists Q'. \ Q \Longrightarrow \text{Cal} \ast Q' \land Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
by blast
next
case (steps P P' P'')
assume \( P \Longrightarrow \text{Cal} \ast P' \) and \( P' \Longrightarrow \text{Cal} \ast P'' \)
hence \( P \Longrightarrow \text{Cal} \ast P'' \)
by (rule steps-add)
thus \( \exists Q'. \ P \Longrightarrow \text{Cal} \ast Q' \land Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P'' \)
using stepsClosure-refl[where Rel=Rel and Cal=Cal]
unfolding refl-on-def
by blast
next
case (trans P Q R)
assume \( \langle P', P \rangle \Longrightarrow \text{Cal} \ast P' \Longrightarrow \exists Q'. \ Q \Longrightarrow \text{Cal} \ast Q' \land Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
and \( P \Longrightarrow \text{Cal} \ast P' \)
from this obtain \( Q' \) where A1: \( Q \Longrightarrow \text{Cal} \ast Q' \) and A2: \( Q' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
by blast
assume \( \langle Q', Q \rangle \Longrightarrow \text{Cal} \ast Q' \Longrightarrow \exists R'. \ R \Longrightarrow \text{Cal} \ast R' \land R' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ Q' \)
with A1 obtain \( R' \) where A3: \( R \Longrightarrow \text{Cal} \ast R' \) and A4: \( R' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ Q' \)
by blast
from A4 A2 have \( R' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
by (rule stepsClosure.trans)
with A3 show \( \exists R'. \ R \Longrightarrow \text{Cal} \ast R' \land R' \rel \Longrightarrow \langle \text{Rel}, \text{Cal} \rangle \ P' \)
by blast
qed
qed

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

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

fixes Rel :: (\'a × \'a) set
and Cal :: \'a processCalculus
assumes sim: weak-reduction-simulation Rel Cal
and con: weak-reduction-contrasimulation (Rel\(^{-1}\)) Cal
shows weak-reduction-coupled-simulation (stepsClosure Rel Cal) Cal
using sim con simulation-and-inverse-contrasimulation-impl-steps-closure-is-contrasimulation
steps-closure-is-weak-reduction-simulation\[where Rel=Rel \text{ and } Cal=Cal\]

by simp

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 \(Rel::('a \times 'a)\) set

and \(Cal::'a\) processCalculus

and \(P Q::'a\)

assumes

\(A1::P \Rightarrow Q, Q \Rightarrow P\)

and \(A2::\text{weak-reduction-simulation}\ Rel\ Cal\)

shows \(\exists Q'. Q \Rightarrow Cal* Q' \land (P, Q') \in Rel^*\)

**proof**

have \(\forall P'. P \Rightarrow Cal* P' \Rightarrow (\exists Q'. Q \Rightarrow Cal* Q' \land (P', Q') \in Rel^*)\)

using \(A1\)

**proof induct**

case \((rel\ P\ Q)\)

assume \((P, Q) \in Rel\)

with \(A2\) have \(\forall P''. P \Rightarrow Cal* P'' \Rightarrow (\exists Q'. Q \Rightarrow Cal* Q' \land (P', Q') \in Rel^*)\)

by blast

thus \(\forall P'. P \Rightarrow Cal* P' \Rightarrow (\exists Q'. Q \Rightarrow Cal* Q' \land (P', Q') \in Rel^*)\)

by blast

**next**

case \((steps\ P\ P')\)

assume \(A::P \Rightarrow Cal* P'\)

show \(\forall P''. P' \Rightarrow Cal* P'' \Rightarrow (\exists Q'. Q \Rightarrow Cal* Q' \land (P'', Q') \in Rel^*)\)

**proof clarify**

fix \(P''\)

assume \(P' \Rightarrow Cal* P''\)

with \(A\) have \(P \Rightarrow Cal* P''\)

by (rule steps-add)

moreover have \((P'', Q'') \in Rel^*\)

by simp

ultimately show \(\exists Q'. P \Rightarrow Cal* Q' \land (P'', Q') \in Rel^*\)

by blast

**qed**

**next**

case \((trans\ P\ Q\ R)\)

assume \(A1::\forall P''. P \Rightarrow Cal* P'' \Rightarrow (\exists Q'. Q \Rightarrow Cal* Q' \land (P', Q') \in Rel^*)\)

and \(A2::\forall Q'. Q \Rightarrow Cal* Q' \Rightarrow (\exists R'. R \Rightarrow Cal* R' \land (Q', R') \in Rel^*)\)

show \(\forall P'. P \Rightarrow Cal* P' \Rightarrow (\exists R'. R \Rightarrow Cal* R' \land (P', R') \in Rel^*)\)

**proof clarify**

fix \(P''\)

assume \(P \Rightarrow Cal* P''\)

with \(A1\) obtain \(Q'\) where \(A3::Q \Rightarrow Cal* Q'\) and \(A4::(P', Q') \in Rel^*\)

by blast

from \(A2\) \(A3\) obtain \(R'\) where \(A5::R \Rightarrow Cal* R'\) and \(A6::(Q', R') \in Rel^*\)

by blast

from \(A4\) \(A6\) have \((P', R') \in Rel^*\)

by simp

with \(A5\) show \(\exists R'. R \Rightarrow Cal* R' \land (P', R') \in Rel^*\)

by blast

**qed**

**qed**

moreover have \(P \Rightarrow Cal* P\)

by (rule steps-refl)

ultimately show \(?thesis\)

by blast

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

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

shows \( \exists Q'. P \mapsto \text{Cal}^* Q' \land (P', Q') \in \text{Rel}^* \)

proof –

have \( \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists Q'. Q \mapsto \text{Cal}^* Q' \land (P', Q') \in \text{Rel}^*) \)

using A1

proof induct

case \((\text{rel} P Q)\)

assume \((P, Q) \in \text{Rel}^{-1}\)

with A2 have \( \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists Q'. Q \mapsto \text{Cal}^* Q' \land (Q', P') \in \text{Rel}^{-1}) \)

by \(\text{blast}\)

thus \( \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists Q'. Q \mapsto \text{Cal}^* Q' \land (P', Q') \in \text{Rel}^*) \)

by \(\text{blast}\)

next

case \((\text{steps} P P')\)

assume A: \( P \mapsto \text{Cal}^* P' \)

show \( \forall P''. P' \mapsto \text{Cal}^* P'' \mapsto (\exists Q'. P \mapsto \text{Cal}^* Q' \land (P'', Q') \in \text{Rel}^*) \)

proof clarify

fix \( P'' \)

with A have \( P \mapsto \text{Cal}^* P'' \)

by \((\text{rule steps-add})\)

moreover have \((P'', P') \in \text{Rel}^*\)

by \(\text{simp}\)

ultimately show \( \exists Q'. P \mapsto \text{Cal}^* Q' \land (P'', Q') \in \text{Rel}^* \)

by \(\text{blast}\)

qed

next

case \((\text{trans} P Q R)\)

assume A1: \( \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists Q'. Q \mapsto \text{Cal}^* Q' \land (P', Q') \in \text{Rel}^*) \)

and A2: \( \forall Q'. Q \mapsto \text{Cal}^* Q' \mapsto (\exists R'. R \mapsto \text{Cal}^* R' \land (Q', R') \in \text{Rel}^*) \)

show \( \forall P'. P \mapsto \text{Cal}^* P' \mapsto (\exists R'. R \mapsto \text{Cal}^* R' \land (P', R') \in \text{Rel}^*) \)

proof clarify

fix \( P' \)

with A1 obtain \( Q' \) where A3: \( Q \mapsto \text{Cal}^* Q' \) and A4: \( (P', Q') \in \text{Rel}^* \)

by \(\text{blast}\)

from A3 A2 obtain \( R' \) where A5: \( R \mapsto \text{Cal}^* R' \) and A6: \( (Q', R') \in \text{Rel}^* \)

by \(\text{blast}\)

from A4 A6 have \((P', R') \in \text{Rel}^*\)

by \(\text{simp}\)

with A5 show \( \exists R'. R \mapsto \text{Cal}^* R' \land (P', R') \in \text{Rel}^* \)

by \(\text{blast}\)

qed

qed

moreover have \( P \mapsto \text{Cal}^* P \)

by \((\text{rule steps-refl})\)

ultimately show ?thesis

by \(\text{blast}\)

qed

end

theory Encodings

imports ProcessCalculi

begin

4 Encodings

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

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

definition STCal
  :: 'procS processCalculus ⇒ 'procT processCalculus
    ⇒ (('procS, 'procT) Proc) processCalculus
where
STCal Source Target ≡
(∀ Reductions = λP P'.
 (∃SP SP', P = SourceTerm SP ∧ P' = SourceTerm SP' ∧ Reductions Source SP SP') ∨
 (∃TP TP', P = TargetTerm TP ∧ P' = TargetTerm TP' ∧ Reductions Target TP TP'))

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

An encoding consists of a source language, a target language, and a mapping from source into target terms.
locale encoding =
  fixes Source :: 'procS processCalculus
    and Target :: 'procT processCalculus
    and Enc :: 'procS ⇒ 'procT

begin
abbreviation enc :: 'procS ⇒ 'procT ([·] [65] 70) where
  [S] ≡ Enc S

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

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

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

abbreviation getTarget
  :: 'procT ⇒ ('procS, 'procT) Proc ⇒ bool (- ∈ T - [70] 80)
where
  T ∈T P ≡ (P = TargetTerm T)

A step of a term in Proc is either a source term step or a target term step.
abbreviation stepST
  :: ('procS, 'procT) Proc ⇒ ('procS, 'procT) Proc ⇒ bool (- →→ ST - [70] 80)
where
  P →→ ST P' ≡
  (∃S S', S ∈S P ∧ S' ∈S P' ∧ S →→ Source S') ∨ (∃T T', T ∈T P ∧ T' ∈T P' ∧ T →→ Target T')

lemma stepST-STCal-step:
  fixes P P' :: ('procS, 'procT) Proc
shows $P \mapsto (STCal \ Source \ Target) \ P' = P \mapsto ST P'$
by (simp add: STCal-def)

lemma STStep-step:
fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) \ Proc$
shows $SourceTerm S \mapsto ST P' = (\exists S'. S' \in S \ P' \land S \mapsto Source S')$
and $TargetTerm T \mapsto ST P' = (\exists T'. T' \in T \ P' \land T \mapsto Target T')$
by blast+

lemma STCal-step:
fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) \ Proc$
shows $SourceTerm S \mapsto (STCal \ Source \ Target) \ P' = (\exists S'. S' \in S \ P' \land S \mapsto Source S')$
and $TargetTerm T \mapsto (STCal \ Source \ Target) \ P' = (\exists T'. T' \in T \ P' \land T \mapsto Target T')$
by (simp add: STCal-def)+

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

abbreviation stepsST :: ('procS, 'procT) \ Proc \Rightarrow ('procS, 'procT) \ Proc \Rightarrow bool ($\mapsto ST^*$ - [70, 70] 80)
where
$P \mapsto ST^* \ P' \equiv \ (\exists S S'. S \in S \ P \land S' \in S \ P' \land S \mapsto Source S') \lor (\exists T T'. T \in T \ P \land T' \in T \ P' \land T \mapsto Target T')$

lemma STSteps-steps:
fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) \ Proc$
shows $SourceTerm S \mapsto ST^* \ P' = (\exists S'. S' \in S \ P' \land S \mapsto Source S')$
and $TargetTerm T \mapsto ST^* \ P' = (\exists T'. T' \in T \ P' \land T \mapsto Target S')$
by blast+

lemma STCal-steps:
fixes $S :: 'procS$
and $T :: 'procT$
and $P' :: ('procS, 'procT) \ Proc$
shows $SourceTerm S \mapsto (STCal \ Source \ Target)^* \ P' = (\exists S'. S' \in S \ P' \land S \mapsto Source S')$
and $TargetTerm T \mapsto (STCal \ Source \ Target)^* \ P' = (\exists T'. T' \in T \ P' \land T \mapsto Target S')$
proof auto
  assume $SourceTerm S \mapsto (STCal \ Source \ Target)^* \ P''$
  from this obtain $n$ where $SourceTerm S \mapsto (STCal \ Source \ Target)^n \ P''$
  by (auto simp add: steps-def)
  thus $\exists S'. S' \in S \ P' \land S \mapsto Source S'$
proof (induct $n$ arbitrary: $P'$)
  case 0
  assume $SourceTerm S \mapsto (STCal \ Source \ Target)^0 \ P'$
  hence $S \in S \ P'$
  by simp
  moreover have $S \mapsto Source S$
  by (rule steps-refl)
  ultimately show $\exists S'. S' \in S \ P' \land S \mapsto Source S'$
  by blast
next
  case (Suc $n \ P''$)
  assume $SourceTerm S \mapsto (STCal \ Source \ Target)^{Suc \ n} \ P''$.
  from this obtain $P'$ where $A1: SourceTerm S \mapsto (STCal \ Source \ Target)^n \ P'$
  and $A2: P' \mapsto (STCal \ Source \ Target) \ P''$
proof

next

assume $\forall P'. \ SourceTerm S \rightarrow (STCal \ Source \ Target)^n \ P' \Rightarrow \exists S'. \ S' \in S \ P' \land S \rightarrow Source \ S'$

with $A1$ obtain $S'$ where $A3: \ S' \in S \ P' \land S \rightarrow Source \ S'$

by auto

from $A2 \ A3$ obtain $S''$ where $A5: \ S'' \in S \ P''$ and $A6: \ S' \rightarrow Source \ S''$

using $STCal-step(1)[where \ S=S' \ and \ P'=P'']$

by blast

from $A4 \ A6$ have $S \rightarrow Source \ S''$

using $step-to-steps[where \ Cal=Source \ and \ P=S' \ and \ P'=S'']$

by $(simp \ add: \ steps-def)$

thus $SourceTerm S \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S')$

proof (induct $n$ arbitrary: $S'$)

next

case $0$

hence $S = S'$

by auto

thus $SourceTerm S \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S')$

by $simp$ add: $steps-refl$

next

case (Suc $n \ S'')$

assume $S \rightarrow Source^{Suc \ n \ S''}$

from $this$ obtain $S'$ where $B1: \ S \rightarrow Source^n \ S' \ and \ B2: \ S' \rightarrow Source \ S''$

by $auto$

assume $\forall S'. \ S \rightarrow Source^n \ S' \Rightarrow SourceTerm S \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S')$

with $B1$ have $SourceTerm S \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S')$

by blast

moreover from $B2$ have $SourceTerm S' \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S'')$

using $step-to-steps[where \ Cal=STCal \ Source \ Target \ and \ P=SourceTerm S']$

by $(simp \ add: \ STCal-def)$

ultimately show $SourceTerm S \rightarrow (STCal \ Source \ Target)^* \ (SourceTerm S'')$

by $(rule \ steps-add)$

qed

next

assume $TargetTerm T \rightarrow (STCal \ Source \ Target)^* \ P'$

from $this$ obtain $n$ where $TargetTerm T \rightarrow (STCal \ Source \ Target)^n \ P'$

by $(auto \ simp \ add: \ steps-def)$

thus $\exists T'. \ T' \in T \ P' \land T \rightarrow Target \ T'$

proof (induct $n$ arbitrary: $P'$)

next

case $0$

hence $T \in T \ P'$

by $simp$

moreover have $T \rightarrow Target \ T$

by $(rule \ steps-refl)$

ultimately show $\exists T'. \ T' \in T \ P' \land T \rightarrow Target \ T'$

by blast

next

case (Suc $n \ P''$)

assume $TargetTerm T \rightarrow (STCal \ Source \ Target)^{Suc \ n \ P''}$

from $this$ obtain $P' \ where \ A1: \ TargetTerm T \rightarrow (STCal \ Source \ Target)^n \ P'$

and $A2: \ P' \rightarrow (STCal \ Source \ Target) \ P''$

by $auto$
assume $\land P', \text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ P' \implies \exists T'. \ T' \in T \ P' \land T \rightarrow \text{Target* } T'$

with $A1$ obtain $T'$ where $A3: \ T' \in T \ P'$ and $A4: \ T \rightarrow \text{Target* } T'$

by blast

from $A2$ $A3$ obtain $T''$ where $A5: \ T'' \in T \ P''$ and $A6: \ T' \rightarrow \text{Target } T''$

using $\text{STCal-step}(2)[\text{where } T=T' \text{ and } P'=P'']$

by blast

from $A4$ $A6$ have $T \rightarrow \text{Target* } T''$

using $\text{step-to-steps}[\text{where } \text{Cal=} \text{Target and } P=T' \text{ and } P'=T'']$

by $(\text{simp add: steps-add}[\text{where } \text{Cal=} \text{Target and } P=T \text{ and } Q=T' \text{ and } R=T''])$

with $A5$ show $\exists T''. \ T'' \in T \ P'' \land T \rightarrow \text{Target* } T''$

by blast

qed

next

fix $T'$

assume $T \rightarrow \text{Target* } T'$

from this obtain $n$ where $T \rightarrow \text{Target}^n \ T'$

by $(\text{auto simp add: steps-def})$

thus $\text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T')$

proof $(\text{induct } n \text{ arbitrary: } T')$

  case $0$
  assume $T \rightarrow \text{Target}^0 \ T'$
  hence $T = T'$
  by auto
  thus $\text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T')$
  by $(\text{simp add: steps-refl})$

next

  case $(\text{Suc } n \ T'')$
  assume $T \rightarrow \text{Target}^{\text{Suc } n} \ T''$
  from this obtain $T'$ where $B1: \ T \rightarrow \text{Target}^n \ T'$ and $B2: \ T' \rightarrow \text{Target } T''$
  by auto
  assume $\land T'. \ T \rightarrow \text{Target}^n \ T' \implies \text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T')$
  with $B1$ have $\text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T')$
  by blast
  moreover from $B2$ have $\text{TargetTerm } T' \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T'')$
  using $\text{step-to-steps}[\text{where } \text{Cal=} \text{STCal Source Target and } P=\text{TargetTerm } T']$
  by $(\text{simp add: ST-Cal-def})$
  ultimately show $\text{TargetTerm } T \rightarrow (\text{STCal Source Target})^n \ (\text{TargetTerm } T'')$
  by $(\text{rule steps-add})$

qed

lemma $\text{stepsST-\text{STCal-steps}}$:
fixes $P \ P' :: (\text{procS}, \text{procT}) \text{ Proc}$
shows $P \rightarrow (\text{STCal Source Target})^n \ P' = P \rightarrow \text{ST* } P'$
proof $(\text{cases } P)$
  case $(\text{SourceTerm } S P)$
  assume $S \in S \ P$
  thus $P \rightarrow (\text{STCal Source Target})^n \ P' = P \rightarrow \text{ST* } P'$
  using $\text{STCal-steps}(1)[\text{where } S=S' \text{ and } P'=P']$ $\text{STSteps-steps}(1)[\text{where } S=S' \text{ and } P'=P']$
  by blast
next
  case $(\text{TargetTerm } TP)$
  assume $TP \in T \ P$
  thus $P \rightarrow (\text{STCal Source Target})^n \ P' = P \rightarrow \text{ST* } P'$
  using $\text{STCal-steps}(2)[\text{where } T=TP \text{ and } P'=P']$ $\text{STSteps-steps}(2)[\text{where } T=TP \text{ and } P'=P']$
  by blast

qed

lemma $\text{stepsST-refl}$:
fixes $P :: (\text{procS}, \text{procT}) \text{ Proc}$
shows \( P \xrightarrow{} ST^* P \)
by (cases \( P \), simp-all add: steps-refl)

lemma stepsST-add:
fixes \( P \, Q \) :: ('procS, 'procT) Proc
assumes A1: \( P \xrightarrow{} ST^* Q \)
and A2: \( Q \xrightarrow{} ST^* R \)
shows \( P \xrightarrow{} ST^* R \)
proof -
from A1 have \( P \xrightarrow{} (STCal Source Target)^* Q \)
by (simp add: stepsST-STCal-steps)
moreover from A2 have \( Q \xrightarrow{} (STCal Source Target)^* R \)
by (simp add: stepsST-STCal-steps)
ultimately have \( P \xrightarrow{} (STCal Source Target)^* R \)
by (rule steps-add)
thus \( P \xrightarrow{} ST^* R \)
by (simp add: stepsST-STCal-steps)
qed

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

abbreviation divergentST
:: ('procS, 'procT) Proc \Rightarrow bool (\xrightarrow{} ST^\omega [70] 80)
where
\( P \xrightarrow{} ST^\omega \equiv (\exists S. S \in S P \land S \xrightarrow{} (Source)^\omega) \lor (\exists T. T \in T P \land T \xrightarrow{} (Target)^\omega) \)

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

lemma divergentST-STCal-divergent:
fixes \( P \) :: ('procS, 'procT) Proc
shows \( P \xrightarrow{} (STCal Source Target)^\omega = P \xrightarrow{} ST^\omega \)
proof (cases \( P \))
case (SourceTerm \( SP \))
assume \( SP \in S P \)
thus \( P \xrightarrow{} (STCal Source Target)^\omega = P \xrightarrow{} ST^\omega \)
using STCal-divergent(1)
by simp
next
case (TargetTerm \( TP \))
assume \( TP \in T P \)
thus \( P \xrightarrow{} (STCal Source Target)^\omega = P \xrightarrow{} ST^\omega \)
using STCal-divergent(2)
by simp
qed

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

abbreviation enc-preserve-pred :: ('procS, 'procT) Proc \Rightarrow bool \Rightarrow bool where
enc-preserved-pred \( Pred \equiv \forall S. Pred \ (SourceTerm \ S) \xrightarrow{} Pred \ (TargetTerm \ ([S])) \)

abbreviation enc-preserve-binary-pred
:: ('procS, 'procT) Proc \Rightarrow 'b \Rightarrow bool
where
enc-preserve-binary-pred \( Pred \equiv \forall S x. Pred \ (SourceTerm \ S) \ x \xrightarrow{} Pred \ (TargetTerm \ ([S]) \ x) \)

An encoding reflects some predicate \( P \) if \( P(S) \) implies \( P(enc S) \) for all source terms \( S \).
**Abbreviation** enc-reflects-pred :: (procS, procT) Proc ⇒ bool ⇒ bool where
e enc-reflects-pred Pred ≡ ∀ S. Pred (TargetTerm ([S])) −→ Pred (SourceTerm S)

**Abbreviation** enc-reflects-binary-pred :: (procS, procT) Proc ⇒ bool ⇒ bool where
e enc-reflects-binary-pred Pred ≡ ∀ S x. Pred (TargetTerm ([S])) x −→ Pred (SourceTerm S)

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

**Abbreviation** enc-respects-pred :: (procS, procT) Proc ⇒ bool ⇒ bool where
e enc-respects-pred Pred ≡ enc-preserves-pred Pred ∧ enc-reflects-pred Pred

**Abbreviation** enc-respects-binary-pred :: (procS, procT) Proc ⇒ bool ⇒ bool where
e enc-respects-binary-pred Pred ≡ enc-preserves-binary-pred Pred ∧ enc-reflects-binary-pred Pred

end

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

**Locale** encoding-wrt-barbs =
  encoding Source Target Enc
  for Source :: procS processCalculus
  and Target :: procT processCalculus
  and Enc :: procS ⇒ procT +
  fixes SWB :: (procS, barbs) calculusWithBarbs
  and TWB :: (procT, barbs) calculusWithBarbs
  assumes calS: calculusWithBarbs.Calculus SWB = Source
  and calT: calculusWithBarbs.Calculus TWB = Target
begin

**Lemma** STCalWB-STCal:
  shows Calculus (STCalWB SWB TWB) = STCal Source Target
  unfolding STCalWB-def using calS calT
  by auto

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

**Abbreviation** hasBarbST :: (procS, procT) Proc ⇒ bool ⇒ bool where
P ↓. a ≡ (∃ S. S ∈ S P ∧ S ↓<SWB>a) ∨ (∃ T. T ∈ T P ∧ T ↓<TWB>a)

**Lemma** STCalWB-hasBarbST:
  fixes P :: (procS, procT) Proc
  and a :: barbs
  shows P ↓<STCalWB SWB TWB>a = P ↓. a
  by (simp add: STCalWB-def)

**Lemma** preservation-of-barbs-in-barbed-encoding:
  fixes Rel :: (procS, procT) Proc × (procS, procT) Proc set
  and P Q :: (procS, procT) Proc
  and a :: barbs
  assumes preservation: rel-preserves-barbs Rel (STCalWB SWB TWB)
  and rel: (P, Q) ∈ Rel
  and barb: P ↓. a
  shows Q ↓. a
using preservation rel barb
by (simp add: STCalWB-def)

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

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

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

abbreviation reachesBarbST :: ('procS, 'procT) Proc ⇒ 'barbs ⇒ bool (-↓, - [70, 70] 80)
where
P ↓ a ≡ (∃ S. S ∈ S P ∧ S ↓ SWB > a) ∨ (∃ T. T ∈ T P ∧ T ↓ TWB > a)

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

lemma weak-preservation-of-barbs-in-barbed-encoding:
fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and P Q :: ('procS, 'procT) Proc
and a :: 'barbs
assumes preservation: rel-weakly-preserves-barbs Rel (STCalWB SWB TWB)
and rel: (P, Q) ∈ Rel
and barb: P ↓ a
shows Q ↓ a
proof
from barb have P ↓ STCalWB SWB TWB > a
  by (simp add: STCalWB-reachesBarbST)
with preservation rel have Q ↓ STCalWB SWB TWB > a
5 Relation between Source and Target Terms

5.1 Relations Induced by the Encoding Function

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

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

by blast
thus Qₜ.a
  by (simp add: STCalWB-reachesBarbST)
qed

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

lemma weak-respection-of-barbs-in-barbed-encoding:
  fixes Rel :: ((procS, procT) Proc × (procS, procT) Proc) set
  and P Q :: (procS, procT) Proc
  and a :: 'barbs
  assumes respection: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
  and rel: (P, Q) ∈ Rel
  shows Pₜ.a = Qₜ.a
proof (rule iffI)
  assume Pₜ.a
  with respection rel show Qₜ.a
    using weak-preservation-of-barbs-in-barbed-encoding[where Rel=Rel]
    by blast
  next
    assume Qₜ.a
    with respection rel show Pₜ.a
      using weak-reflection-of-barbs-in-barbed-encoding[where Rel=Rel]
      by blast
qed
abbreviation (in encoding) \textit{indRelR}\textit{ infix} ::
\[\langle \text{procS, procT} \rangle \text{ Proc} \Rightarrow \langle \text{procS, procT} \rangle \text{ Proc} \Rightarrow \text{ bool} \ (- [\cdot] R - [75, 75] 80)\]
where
\[P \circ [-\ R Q \equiv (P, Q) \in \text{indRelR}\]

inductive-set (in encoding) \textit{indRelRPO} :: ((\langle \text{procS, procT} \rangle \text{ Proc}) \times (\langle \text{procS, procT} \rangle \text{ Proc})) set
where
\[\text{encR: (SourceTerm S, TargetTerm ([S]))} \in \text{indRelRPO} | \]
\[\text{source: (SourceTerm S, SourceTerm S) } \in \text{indRelRPO} | \]
\[\text{target: (TargetTerm T, TargetTerm T) } \in \text{indRelRPO} | \]
\[\text{trans: } [[(P, Q) \in \text{indRelRPO}; (Q, R) \in \text{indRelRPO}] \Rightarrow (P, R) \in \text{indRelRPO}\]

abbreviation (in encoding) \textit{indRelRPO}\textit{ infix} ::
\[\langle \text{procS, procT} \rangle \text{ Proc} \Rightarrow \langle \text{procS, procT} \rangle \text{ Proc} \Rightarrow \text{ bool} \ (- \lesssim [\cdot] R - [75, 75] 80)\]
where
\[P \lesssim [\cdot] R Q \equiv (P, Q) \in \text{indRelRPO}\]

lemma (in encoding) \textit{indRelRPO-refl}: shows refl \textit{indRelRPO}
unfolding refl-on-def
proof auto
fix P
show \(P \lesssim [\cdot] R P\)
proof (cases P)
case (SourceTerm SP)
assume SP \(\in S\) P
thus \(P \lesssim [\cdot] R P\)
by (simp add: \textit{indRelRPO}.source)
next
case (TargetTerm TP)
assume TP \(\in T\) P
thus \(P \lesssim [\cdot] R P\)
by (simp add: \textit{indRelRPO}.target)
qed
qed

lemma (in encoding) \textit{indRelRPO-is-preorder}: shows preorder \textit{indRelRPO}
unfolding preorder-on-def
proof
show refl \textit{indRelRPO}
by (rule \textit{indRelRPO-refl})
next
show trans \textit{indRelRPO}
unfolding trans-def
proof clarify
fix P Q R
assume \(P \lesssim [\cdot] R Q \text{ and } Q \lesssim [\cdot] R R\)
thus \(P \lesssim [\cdot] R R\)
by (rule \textit{indRelRPO}.trans)
qed
qed

lemma (in encoding) \textit{refl-trans-closure-of-indRelR}: shows \textit{indRelRPO} = \textit{indRelR}\ast
proof auto
fix P Q
assume \(P \lesssim [\cdot] R Q\)
thus \((P, Q) \in \text{indRelR}\ast\)
proof induct
case (encR S)
  show (SourceTerm S, TargetTerm (\[ S \])) \in indRelR^*
    using indRelR.encR[of S]
    by simp
next
case (source S)
  show (SourceTerm S, SourceTerm S) \in indRelR^*
    by simp
next
case (target T)
  show (TargetTerm T, TargetTerm T) \in indRelR^*
    by simp
next
case (trans P Q R)
  assume (P, Q) \in indRelR^* and (Q, R) \in indRelR^*
  thus (P, R) \in indRelR^*
    by simp
qed

next
fix P Q
assume (P, Q) \in indRelR^*
thus P \preceq_{\[ \cdot \]} \Rightarrow R Q
proof induct
  show P \preceq_{\[ \cdot \]} \Rightarrow R P
    using indRelRPO-refl
    unfolding refl-on-def
    by simp
next
case (step Q R)
  assume P \preceq_{\[ \cdot \]} \Rightarrow R Q
  moreover assume Q \preceq_{\[ \cdot \]} \Rightarrow R R
  hence Q \preceq_{\[ \cdot \]} \Rightarrow R R
    by (induct, simp add: indRelRPO.encR)
  ultimately show P \preceq_{\[ \cdot \]} \Rightarrow R R
    by (rule indRelRPO.trans)
qed

qed

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

lemma (in encoding) indRelR-impl-exists-source-target-relation:
  fixes PredA :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set \Rightarrow bool
  and PredB :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) \Rightarrow bool
  shows PredA indRelR \Longrightarrow \exists Rel. (\forall S. (SourceTerm S, TargetTerm (\[ S \])) \in Rel) \land PredA Rel
  and \forall (P, Q) \in indRelR. PredB (P, Q)
    \Longrightarrow \exists Rel. (\forall S. (SourceTerm S, TargetTerm (\[ S \])) \in Rel) \land (\forall (P, Q) \in Rel. PredB (P, Q))
proof -
  have A: \forall S. SourceTerm S \preceq_{\[ \cdot \]} \Rightarrow TargetTerm (\[ S \])
    by (simp add: indRelR.encR)
  thus PredA indRelR \Longrightarrow \exists Rel. (\forall S. (SourceTerm S, TargetTerm (\[ S \])) \in Rel) \land PredA Rel
    by blast
  with A show \forall (P, Q) \in indRelR. PredB (P, Q)
    \Longrightarrow \exists Rel. (\forall S. (SourceTerm S, TargetTerm (\[ S \])) \in Rel) \land (\forall (P, Q) \in Rel. PredB (P, Q))
    by blast
qed

lemma (in encoding) source-target-relation-impl-indRelR:
  fixes Rel :: (('procS, 'procT) Proc \times ('procS, 'procT) Proc) set
and \( \text{Pred} :: (\langle \text{procS}, \text{procT} \rangle \text{Proc} \times (\langle \text{procS}, \text{procT} \rangle \text{Proc})) \Rightarrow \text{bool} \)

**proof clarify**

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

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

**proof**

**induct**

**thus** \( \text{Pred} (P, Q) \)

**case** \( \text{encR} S \)

**have** \( \text{SourceTerm} S \mathcal{R} \text{TargetTerm} ([S]) \)

**by** \( \text{simp add: indRelR.encR} \)

**with** \( A \) **show** \( \text{Pred} (\text{SourceTerm} S, \text{TargetTerm} ([S])) \)

**by** \( \text{simp} \)

**next**

**case** \( \text{source} S \)

**from** \( \text{reflCond} \) **show** \( \text{Pred} (\text{SourceTerm} S, \text{SourceTerm} S) \)

**by** \( \text{simp} \)

**next**

**case** \( \text{target} T \)

**from** \( \text{reflCond} \) **show** \( \text{Pred} (\text{TargetTerm} T, \text{TargetTerm} T) \)

**by** \( \text{simp} \)

**next**

**case** \( \text{trans} P Q R \)

**assume** \( \text{Pred} (P, Q) \) **and** \( \text{Pred} (Q, R) \)

**with** \( \text{transCond} \) **show** \( \text{Pred} (P, R) \)

**by** \( \text{blast} \)

**qed**

**next**

**fix** \( P Q \)

**assume** \( \forall x \in \text{indRelRPO}. \text{Pred} x \) **and** \( P \mathcal{R} \text{TargetTerm} ([S]) \)

**thus** \( \text{Pred} (P, Q) \)

**by** \( \text{auto simp add: indRelRPO.encR indRelR.simps} \)

**qed**

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

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

**assumes** \( \text{reflCond} : \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \)

**and** \( \text{transCond} : \forall P Q R. (\text{Pred} (P, Q) \land \text{Pred} (Q, R)) \Rightarrow \text{Pred} (P, R) \)

**shows** \( \forall (P, Q) \in \text{indRelRPO}. \text{Pred} (P, Q) \)
∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel)

proof (rule iffI)
have ∀ S. SourceTerm S ≤R TargetTerm ([S])
  by (simp add: indRelRPO, encR)
moreover have preorder indRelRPO
  using indRelRPO-is-preorder
  by blast
moreover assume ∀ (P, Q) ∈ indRelRPO. Pred (P, Q)
ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel
  by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ (P, Q) ∈ Rel. Pred (P, Q)) ∧ preorder Rel
from this obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  and A2: ∀ (P, Q) ∈ Rel. Pred (P, Q) and A3: preorder Rel
  by blast
show ∀ (P, Q) ∈ indRelRPO. Pred (P, Q)
proof clarify
fix P Q
assume P ≤R Q
hence (P, Q) ∈ Rel
proof induct
case (encR S)
  from A1 show (SourceTerm S, TargetTerm ([S])) ∈ Rel
    by simp
next
case (source S)
  from A3 show (SourceTerm S, SourceTerm S) ∈ Rel
    unfolding preorder-on-def refl-on-def
    by simp
next
case (target T)
  from A3 show (TargetTerm T, TargetTerm T) ∈ Rel
    unfolding preorder-on-def refl-on-def
    by simp
next
case (trans P Q R)
  assume (P, Q) ∈ Rel and (Q, R) ∈ Rel
  with A3 show (P, R) ∈ Rel
    unfolding preorder-on-def trans-def
    by blast
qed
with A2 show Pred (P, Q)
  by simp
qed
qed

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

lemma (in encoding) enc-satisfies-pred-impl-indRelR-satisfies-pred:
  fixes Pred :: ('procS, 'procT) Proc × ('procS, 'procT) Proc ⇒ bool
  assumes encCond: ∀ S. Pred (SourceTerm S, TargetTerm ([S]))
  shows ∀ (P, Q) ∈ indRelR. Pred (P, Q)
    by (auto simp add: encCond indRelR.simps)

lemma (in encoding) indRelR-satisfies-pred-impl-enc-satisfies-pred:
  fixes Pred :: ('procS, 'procT) Proc × ('procS, 'procT) Proc ⇒ bool
  assumes relCond: ∀ (P, Q) ∈ indRelR. Pred (P, Q)
  shows ∀ S. Pred (SourceTerm S, TargetTerm ([S]))
lemma (in encoding) enc-satisfies-pred-iff-indRelR-satisfies-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows (∀ S. Pred (SourceTerm S, TargetTerm ([S])) = (∀ (P, Q) ∈ indRelR. Pred (P, Q))
  using enc-satisfies-pred-impl-indRelR-satisfies-pred[where Pred=Pred]
  by simp

lemma (in encoding) enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-satisfies-binary-pred Pred = rel-satisfies-binary-pred indRelR Pred
  using enc-satisfies-binary-pred-iff-indRelR-satisfies-binary-pred[where Pred=λ(P, Q) a. Pred P a → Pred Q a]
  by simp

lemma (in encoding) enc-preserves-pred-iff-indRelPO-preserves-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-preserves-pred Pred = rel-preserves-pred indRelR PO Pred
  using enc-preserves-pred-iff-indRelPO-preserves-pred[where Pred=Pred]
  by simp

lemma (in encoding) enc-preserves-binary-pred-iff-indRelPO-preserves-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-preserves-binary-pred Pred = rel-preserves-binary-pred indRelR PO Pred
  using enc-preserves-binary-pred-iff-indRelPO-preserves-binary-pred[where Pred=λ(P, Q) a. Pred P a → Pred Q a]
  by simp

lemma (in encoding) enc-reflects-pred-iff-indRelR-reflects-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-reflects-pred Pred = rel-reflects-pred indRelR Pred
  using enc-reflects-pred-iff-indRelR-reflects-pred[where Pred=Pred]
  by simp

lemma (in encoding) enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-reflects-binary-pred Pred = rel-reflects-binary-pred indRelR Pred
  using enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred[where Pred=λ(P, Q) a. Pred P a → Pred Q a]
  by simp

lemma (in encoding) enc-reflects-pred-iff-indRelRPO-reflects-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-reflects-pred Pred = rel-reflects-pred indRelR PO Pred
  using enc-reflects-pred-iff-indRelRPO-reflects-pred[where Pred=Pred]
  by simp

lemma (in encoding) enc-reflects-binary-pred-iff-indRelRPO-reflects-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-reflects-binary-pred Pred = rel-reflects-binary-pred indRelR PO Pred
  using enc-reflects-binary-pred-iff-indRelRPO-reflects-binary-pred[where Pred=λ(P, Q) a. Pred P a → Pred Q a]
  by simp

lemma (in encoding) enc-reflects-pred-iff-indRelR-respects-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-reflects-pred Pred = rel-reflects-pred indRelR Pred
using \( 	ext{enc-preserves-pred-iff-indRelR-preserves-pred} \) where \( \text{Pred} = \text{Pred} \)
\( \text{enc-reflects-pred-iff-indRelR-reflects-pred} \) where \( \text{Pred} = \text{Pred} \)

by blast

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

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

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

using \( \text{enc-preserves-binary-pred-iff-indRelR-preserves-binary-pred} \) where \( \text{Pred} = \text{Pred} \)
\( \text{enc-reflects-binary-pred-iff-indRelR-reflects-binary-pred} \) where \( \text{Pred} = \text{Pred} \)

by blast

**lemma (in encoding) enc-respects-pred-iff-indRelRPO-respects-pred:**

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

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

using \( \text{enc-respects-pred-iff-indRelRPO-respects-pred} \) where \( \text{Pred} = \text{Pred} \)

apply simp by blast

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

**lemma (in encoding) enc-satisfies-pred-iff-source-target-satisfies-pred:**

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

**shows** \( \forall \text{Pred} \in \text{\textquoteleft.bool} \) and \( \forall \text{Pred} \in \text{\textquoteleft.bool} \)

using \( \text{enc-satisfies-pred-iff-indRelR-satisfies-pred} \) where \( \text{Pred} = \text{Pred} \)
\( \text{indRelR-iff-exists-source-target-relation} \) where \( \text{Pred} = \text{Pred} \)

by simp

next

have \( \forall \text{Pred} \in \text{\textquoteleft.bool} \)

using \( \text{enc-satisfies-pred-iff-indRelR-satisfies-pred} \) where \( \text{Pred} = \text{Pred} \)

by simp

moreover assume \( \forall \text{Pred} \in \text{\textquoteleft.bool} \)

hence \( \forall \text{Pred} \in \text{\textquoteleft.bool} \)

by blast

ultimately show \( \forall \text{Pred} \in \text{\textquoteleft.bool} \)

using \( \text{indRelRPO-iff-exists-source-target-relation} \) where \( \text{Pred} = \text{Pred} \)

by simp

qed

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

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

**shows** \( \text{enc-preserves-pred Pred} = (\exists \text{Rel} \in \text{\textquoteleft.bool} \) and \( \text{rel-preserves-pred Rel Pred} \)

and \( \text{enc-preserves-pred Pred} = (\exists \text{Rel} \in \text{\textquoteleft.bool} \) and \( \text{rel-preserves-pred Rel Pred} \)

by blast

moreover have \( A2 : \forall \text{Rel} \in \text{\textquoteleft.bool} \)

by blast
ultimately show enc-preserves-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-preserves-pred Rel Pred)

using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), Pred P → Pred Q]
by simp

from A1 A2 show enc-preserves-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-preserves-pred Rel Pred ∧ preorder Rel)
using enc-satisfies-pred-iff-source-target-satisfies-pred(2)[where
Pred=λ(P, Q), Pred P → Pred Q]
by simp
qed

lemma (in encoding) enc-preserves-binary-pred-iff-source-target-rel-preserves-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
shows enc-preserves-binary-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-preserves-binary-pred Rel Pred)
proof –
have enc-preserves-binary-pred Pred
= (∀ S. (λ(P, Q), ∀ a. Pred P a → Pred Q a) (SourceTerm S, TargetTerm ([S])))
by blast

moreover have \rel. rel-preserves-binary-pred Rel Pred
= (∀ (P, Q) ∈ Rel. (λ(P, Q), ∀ a. Pred P a → Pred Q a) (P, Q))
by blast

ultimately show enc-preserves-binary-pred Pred = (∃ Rel. (∀ S. 
(SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-preserves-binary-pred Rel Pred)

using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), ∀ a. Pred P a → Pred Q a]
by simp
qed

lemma (in encoding) enc-reflects-pred-iff-source-target-rel-reflects-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
shows enc-reflects-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-reflects-pred Rel Pred ∧ preorder Rel)
proof –
have A1: enc-reflects-pred Pred
= (∀ S. (λ(P, Q), Pred Q → Pred P) (SourceTerm S, TargetTerm ([S])))
by blast

moreover have A2: \rel. rel-reflects-pred Rel Pred
= (∀ (P, Q) ∈ Rel. (λ(P, Q), Pred Q → Pred P) (P, Q))
by blast

ultimately show enc-reflects-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-reflects-pred Rel Pred)

using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), Pred Q → Pred P]
by simp

from A1 A2 show enc-reflects-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-reflects-pred Rel Pred ∧ preorder Rel)
using enc-satisfies-pred-iff-source-target-satisfies-pred(2)[where
Pred=λ(P, Q), Pred Q → Pred P]
by simp
qed

lemma (in encoding) enc-reflects-binary-pred-iff-source-target-rel-reflects-binary-pred:
fixes Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
shows enc-reflects-binary-pred Pred = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel) 
∧ rel-reflects-binary-pred Rel Pred)
proof –
have enc-reflects-binary-pred Pred
= (∀ S. (λ(P, Q). ∀ a. Pred Q a → Pred P a) (SourceTerm S, TargetTerm ([S])))
by blast

moreover have \( \bigwedge \text{Rel. rel-reflects-binary-pred Rel Pred} \)
= (∀ (P, Q) ∈ Rel. (λ(P, Q). ∀ a. Pred Q a → Pred P a) (P, Q))
by blast

ultimately show enc-reflects-binary-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-reflects-binary-pred Rel Pred
using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), ∀ a. Pred Q a → Pred P a]
by simp

qed

Lemma (in encoding) enc-respects-pred-iff-source-target-rel-respects-pred-encR:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-respects-pred Pred
= (∃ S. (λ(P, Q). Pred P = Pred Q) (SourceTerm S, TargetTerm ([S])))
and enc-respects-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel Pred ∧ preorder Rel

proof –

have A1: enc-respects-pred Pred
= (∀ S. (λ(P, Q). Pred P = Pred Q) (SourceTerm S, TargetTerm ([S])))
by blast

moreover have A2: \( \bigwedge \text{Rel. rel-respects-pred Rel Pred} \) = (∀ (P, Q) ∈ Rel. (λ(P, Q). Pred P = Pred Q) (P, Q))
by blast

ultimately show enc-respects-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel Pred
using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), Pred P = Pred Q]
by simp

from A1 A2 show enc-respects-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ preorder Rel
using enc-satisfies-pred-iff-source-target-satisfies-pred(2)[where
Pred=λ(P, Q), Pred P = Pred Q]
by simp

qed

Lemma (in encoding) enc-respects-binary-pred-iff-source-target-rel-respects-binary-pred-encR:
fixes Pred :: ('procS, 'procT) Proc ⇒ bool
shows enc-respects-binary-pred Pred
= (∃ S. (λ(P, Q). ∀ a. Pred P a = Pred Q a) (SourceTerm S, TargetTerm ([S])))
and enc-respects-binary-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-binary-pred Rel Pred

proof –

have enc-respects-binary-pred Pred
= (∀ S. (λ(P, Q). ∀ a. Pred P a = Pred Q a) (SourceTerm S, TargetTerm ([S])))
by blast

moreover have \( \bigwedge \text{Rel. rel-respects-binary-pred Rel Pred} \) = (∀ (P, Q) ∈ Rel. (λ(P, Q). ∀ a. Pred P a = Pred Q a) (P, Q))
by blast

ultimately show enc-respects-binary-pred Pred = (∃ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-binary-pred Rel Pred
using enc-satisfies-pred-iff-source-target-satisfies-pred(1)[where
Pred=λ(P, Q), ∀ a. Pred P a = Pred Q a]
by simp

qed

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

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

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

inductive-set (in encoding) indRelLPO
  where
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelLPO |
  source: (SourceTerm S, SourceTerm S) ∈ indRelLPO |
  target: (TargetTerm T, TargetTerm T) ∈ indRelLPO |
  trans: [(P, Q) ∈ indRelLPO; (Q, R) ∈ indRelLPO] ⇒ (P, R) ∈ indRelLPO

abbreviation (in encoding) indRelLPOinfix ::
  where
  P ≲[[]]L Q ≡ (P, Q) ∈ indRelLPO

lemma (in encoding) indRelLPO-refl:
  shows refl indRelLPO
  unfolding refl-on-def
proof auto
fix P
show P ≲[[]]L P
proof (cases P)
case (SourceTerm SP)
  assume SP ∈ S P
  thus P ≲[[]]L P
  by (simp add: indRelLPO.source)
next
case (TargetTerm TP)
  assume TP ∈ T P
  thus P ≲[[]]L P
  by (simp add: indRelLPO.target)
qed
qed

lemma (in encoding) indRelLPO-is-preorder:
  shows preorder indRelLPO
  unfolding preorder-on-def
proof
  show refl indRelLPO
    by (rule indRelLPO-refl)
next
  show trans indRelLPO
    unfolding trans-def
proof clarify
fix P Q R
  assume P ≲[[]]L Q and Q ≲[[]]L R
  thus P ≲[[]]L R
    by (rule indRelLPO.trans)
qed
qed

lemma (in encoding) refl-trans-closure-of-indRelL:
  shows indRelLPO = indRelL
proof auto
fix P Q
  assume P ≲[[]]L Q
thus \((P, Q) \in \text{indRelL}^*\)
proof induct
  case \((\text{encL } S)\)
  show \((\text{TargetTerm } [S], \text{SourceTerm } S) \in \text{indRelL}^*\)
  using \text{indRelL}L\text{encL}(S)
  by simp
next
  case \((\text{source } S)\)
  show \((\text{SourceTerm } S, \text{SourceTerm } S) \in \text{indRelL}^*\)
  by simp
next
  case \((\text{target } T)\)
  show \((\text{TargetTerm } T, \text{TargetTerm } T) \in \text{indRelL}^*\)
  by simp
next
  case \((\text{trans } P Q R)\)
  assume \((P, Q) \in \text{indRelL}^*\) and \((Q, R) \in \text{indRelL}^*\)
thus \((P, R) \in \text{indRelL}^*\)
  by simp
qed
next
fix \(P Q\)
assume \((P, Q) \in \text{indRelL}^*\)
thus \(P \preceq L Q\)
proof induct
  show \(P \preceq L P\)
  using \text{indRelL}L\text{po-refl}
  unfolding \text{refl-on-def}
  by simp
next
  case \((\text{step } Q R)\)
  assume \(P \preceq L Q\)
  moreover assume \(Q \preceq L R\)
  hence \(Q \preceq L R\)
  by \((\text{induct}, \text{simp add: indRelLpo}\text{encL})\)
  ultimately show \(P \preceq L R\)
  by \((\text{simp add: indRelLpo}\text{trans}[of \ P \ Q \ \ R])\)
qed

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

lemma \text{(in encoding)} \text{indRelR-preserves-pred-iff-indRelL-reflects-pred}:
  fixes \text{Pred} :: \((\text{procS}, \text{procT}) \Rightarrow \text{Proc} \Rightarrow \text{bool}\)
  shows \text{rel-preserves-pred indRelR}\text{Pred} = \text{rel-reflects-pred indRelL}\text{Pred}
proof
  assume \text{preservation}
  show \text{rel-reflects-pred indRelR}\text{Pred}
  proof clarify
    fix \(P Q\)
    assume \(P \preceq L Q\)
    from this obtain \(S\) where \(S \in S Q\) and \([S] \in T P\)
    by \((\text{induct}, \text{blast})\)
    hence \(Q \preceq R P\)
    by \((\text{simp add: indRelR}\text{encR})\)
    moreover assume \(\text{Pred} Q\)
    ultimately show \(\text{Pred} P\)
      using \text{preservation}
      by \text{blast}
qed

next

assume reflection: rel-reflects-pred indRelL Pred
show rel-preserves-pred indRelR Pred

proof clarify
fix P Q
assume P \text{ rel-reflects-pred} indRelL Q
from this obtain S where S \subseteq S P and [S] \subseteq T Q
by (induct, blast)
hence Q \text{ rel-preserves-pred} indRelR P
by (simp add: indRelL.encL)
moreover assume Pred P
ultimately show Pred Q
using reflection
by blast
qed

qed

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

proof
assume preservation: rel-preserves-binary-pred indRelR Pred
show rel-reflects-binary-pred indRelL Pred

proof clarify
fix P Q x
assume P \text{ rel-preserves-binary-pred} indRelR Q
from this obtain S where S \subseteq S Q and [S] \subseteq T P
by (induct, blast)
hence Q \text{ rel-reflects-binary-pred} indRelL P
by (simp add: indRelL.encR)
moreover assume Pred Q x
ultimately show Pred P x
using preservation
by blast
qed

next

assume reflection: rel-reflects-binary-pred indRelL Pred
show rel-preserves-binary-pred indRelR Pred

proof clarify
fix P Q x
assume P \text{ rel-reflects-binary-pred} indRelL Q
from this obtain S where S \subseteq S P and [S] \subseteq T Q
by (induct, blast)
hence Q \text{ rel-preserves-binary-pred} indRelR P
by (simp add: indRelL.encL)
moreover assume Pred P x
ultimately show Pred Q x
using reflection
by blast
qed

qed

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

proof
assume reflection: rel-reflects-pred indRelR Pred
show rel-preserves-pred indRelL Pred

proof clarify
fix P Q
assume \( P \, \mathcal{R} L \, Q \)
from this obtain \( S \) where \( S \in S \, Q \) and \( [S] \in T \, P \)
by (induct, blast)
hence \( Q \, \mathcal{R} R \, P \)
by (simp add: indRelR.encR)
moreover assume \( \text{Pred} \, P \)
ultimately show \( \text{Pred} \, Q \)
using reflection
by blast
qed
next
assume preservation: rel-preserves-pred indRelL \, \text{Pred}
show rel-reflects-pred indRelR \, \text{Pred}
proof clarify
fix \( P \, Q \)
assume \( P \, \mathcal{R} R \, Q \)
from this obtain \( S \) where \( S \in S \, P \) and \( [S] \in T \, Q \)
by (induct, blast)
hence \( Q \, \mathcal{R} L \, P \)
by (simp add: indRelL.encL)
moreover assume \( \text{Pred} \, P \) \, \text{x}
ultimately show \( \text{Pred} \, Q \) \, \text{x}
using preservation
by blast
qed
qed
lemma (in encoding) indRelR-reflects-binary-pred-iff-indRelL-preserves-binary-pred:
fixes \( \text{Pred} :: (\text{proc}S, \text{proc}T) \, \text{Proc} \Rightarrow 'b \Rightarrow \text{bool} \)
shows rel-reflects-binary-pred indRelR \, \text{Pred} = rel-preserves-binary-pred indRelL \, \text{Pred}
proof
assume reflection: rel-reflects-binary-pred indRelR \, \text{Pred}
show rel-preserves-binary-pred indRelL \, \text{Pred}
proof clarify
fix \( P \, Q \, x \)
assume \( P \, \mathcal{R} L \, Q \)
from this obtain \( S \) where \( S \in S \, P \) and \( [S] \in T \, Q \)
by (induct, blast)
hence \( Q \, \mathcal{R} R \, P \)
by (simp add: indRelL.encL)
moreover assume \( \text{Pred} \, P \) \, \text{x}
ultimately show \( \text{Pred} \, Q \) \, \text{x}
using reflection
by blast
qed
next
assume preservation: rel-preserves-binary-pred indRelL \, \text{Pred}
show rel-reflects-binary-pred indRelR \, \text{Pred}
proof clarify
fix \( P \, Q \, x \)
assume \( P \, \mathcal{R} R \, Q \)
from this obtain \( S \) where \( S \in S \, P \) and \( [S] \in T \, Q \)
by (induct, blast)
hence \( Q \, \mathcal{R} L \, P \)
by (simp add: indRelL.encL)
moreover assume \( \text{Pred} \, Q \) \, \text{x}
ultimately show \( \text{Pred} \, P \) \, \text{x}
using preservation
by blast
qed
qed
lemma (in encoding) indRelR-respects-pred-iff-indRelL-respects-pred:
  fixes Pred :: ('procS, 'procT) Proc ⇒ bool
  shows rel-respects-pred-indRelR Pred = rel-respects-pred indRelL Pred
  using
proof
  ultimately show from Pred
  have
  by blast
  qed

lemma (in encoding) indRelR-respects-binary-pred-iff-indRelL-respects-binary-pred:
  fixes Pred :: ('procS, 'procT) Proc ⇒ ('b ⇒ bool)
  shows rel-respects-binary-pred-indRelR Pred = rel-respects-binary-pred indRelL Pred
  using
proof
  ultimately show from Pred
  have
  by blast
  qed

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

lemma (in encoding) indRelR-cond-binary-preservation-iff-indRelL-cond-binary-reflection:
  fixes Pred :: ('procS, 'procT) Proc ⇒ ('b ⇒ ('b ⇒ bool))
  shows (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-preserves-binary-pred Rel Pred)
       = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]), SourceTerm S) ∈ Rel)
            ∧ rel-reflects-binary-pred Rel Pred)
proof
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]), SourceTerm S) ∈ Rel) ∧ rel-preserves-binary-pred Rel Pred
  then obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm ([S]), SourceTerm S) ∈ Rel
                  and A2: rel-preserves-binary-pred Rel Pred
  by blast
  from A1 have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel−1
  by simp
  moreover from A2 have rel-reflects-binary-pred (Rel−1) Pred
  by simp
  ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-reflects-binary-pred Rel Pred
  by blast
next
assumption \( \exists S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
then obtain \( R \) where \( B_1: \forall S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
and \( B_2: \) rel-reflects-binary-pred R R
by blast
from \( B_1 \) have \( \forall S. ([S]_S, S) \in R^{-1} \)
by simp
moreover
from \( B_2 \) have \( rel-reflects-binary-pred (R^{-1}) R \)
by simp
ultimately
show \( \exists S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
by blast
\( \text{qed} \)

lemma (in encoding) indRelR-cond-reflection-iff-indRelR-cond-preservation:
fixes Proc :: ('procS, 'procT) Proc \Rightarrow bool
shows \( \exists S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
proof
assumption \( \exists S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
then obtain \( R \) where \( A_1: \forall S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
and \( A_2: \) rel-reflects-pred R R
by blast
from \( A_1 \) have \( \forall S. ([S]_S, S) \in R^{-1} \)
by simp
moreover
from \( A_2 \) have \( rel-reflects-pred (R^{-1}) R \)
by simp
ultimately
show \( \exists S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
by blast
next
assumption \( \exists S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
then obtain \( R \) where \( B_1: \forall S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
and \( B_2: \) rel-reflects-pred R R
by blast
from \( B_1 \) have \( \forall S. ([S]_S, S) \in R^{-1} \)
by simp
moreover
from \( B_2 \) have \( rel-reflects-pred (R^{-1}) R \)
by simp
ultimately
show \( \exists S. ([S]_S, S) \in R \land rel-reflects-pred R R \)
by blast
\( \text{qed} \)

lemma (in encoding) indRelR-cond-binary-reflection-iff-indRelR-cond-binary-preservation:
fixes Proc :: ('procS, 'procT) Proc \Rightarrow bool
shows \( \exists S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
proof
assumption \( \exists S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
then obtain \( R \) where \( A_1: \forall S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
and \( A_2: \) rel-reflects-binary-pred R R
by blast
from \( A_1 \) have \( \forall S. ([S]_S, S) \in R^{-1} \)
by simp
moreover
from \( A_2 \) have \( rel-reflects-binary-pred (R^{-1}) R \)
by simp
ultimately
show \( \exists S. ([S]_S, S) \in R \land rel-reflects-binary-pred R R \)
by blast
\( \text{qed} \)
next
assume \( \exists \text{Rel. } (\forall S. (\text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \text{Rel}) \) \& \( \text{rel-preserves-binary-pred Rel Pred} \)
then obtain \( \text{Rel where } B1: \forall S. (\text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \text{Rel} \)
and \( B2: \text{rel-preserves-binary-pred Rel Pred} \)

by blast
from \( B1 \) have \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel} \)
by simp
moreover from \( B2 \) have \( \text{rel-reflects-binary-pred } (\text{Rel}^{-1}) \) \( \text{Pred} \)
by simp
ultimately show \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-reflects-binary-pred Rel Pred} \)
by blast
qed

lemma (in encoding) \( \text{indRelR-cond-respection-iff-indRelL-cond-respection:} \)
\begin{align*}
\text{fixes } & \text{Pred } :: \langle \text{’procS, ’procT} \rangle \text{Proc } \Rightarrow \text{bool} \\
\text{shows } & (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \& \text{rel-refsects-binary-pred Rel Pred}) \\
= & (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{SourceTerm } S) \in \text{Rel}) \& \text{rel-refsects-binary-pred Rel Pred})
\end{align*}

proof
assume \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
from this obtain \( \text{Rel where } A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel} \)
and \( A2: \text{rel-refsects-binary-pred Rel Pred} \)

by blast
from \( A1 \) have \( \forall S. (\text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \{(a, b). \ (b, a) \in \text{Rel}\}\)
by simp
moreover from \( A2 \) have \( \text{rel-refsects-binary-pred } \{(a, b). \ (b, a) \in \text{Rel}\} \) \( \text{Pred} \)
by blast
ultimately show \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
by blast
next
assume \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
from this obtain \( \text{Rel where } A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \text{Rel} \)
and \( A2: \text{rel-refsects-binary-pred Rel Pred} \)

by blast
from \( A1 \) have \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \{(a, b). \ (b, a) \in \text{Rel}\}\)
by simp
moreover from \( A2 \) have \( \text{rel-refsects-binary-pred } \{(a, b). \ (b, a) \in \text{Rel}\} \) \( \text{Pred} \)
by blast
ultimately show \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
by blast
qed

lemma (in encoding) \( \text{indRelR-cond-binary-respection-iff-indRelL-cond-binary-respection:} \)
\begin{align*}
\text{fixes } & \text{Pred } :: \langle \text{’procS, ’procT} \rangle \text{Proc } \Rightarrow \text{bool} \\
\text{shows } & (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \& \text{rel-refsects-binary-pred Rel Pred}) \\
= & (\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{SourceTerm } S) \in \text{Rel}) \& \text{rel-refsects-binary-pred Rel Pred})
\end{align*}

proof
assume \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
from this obtain \( \text{Rel where } A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel} \)
and \( A2: \text{rel-refsects-binary-pred Rel Pred} \)

by blast
from \( A1 \) have \( \forall S. (\text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \{(a, b). \ (b, a) \in \text{Rel}\}\)
by simp
moreover from \( A2 \) have \( \text{rel-refsects-binary-pred } \{(a, b). \ (b, a) \in \text{Rel}\} \) \( \text{Pred} \)
by blast
ultimately show \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } \langle [S] \rangle) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
by blast
next
assume \( \exists \text{Rel. } (\forall S. (\text{TargetTerm } \langle [S] \rangle, \text{SourceTerm } S) \in \text{Rel}) \) \& \( \text{rel-refsects-binary-pred Rel Pred} \)
from this obtain \( \forall S. (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel} \) and \( A2: \text{rel-respects-binary-pred Rel Pred} \)

by \text{blast}

from \( A1 \) have \( \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \{(a, b). (b, a) \in \text{Rel}\} \)

by \text{simp}

moreover from \( A2 \) have \( \text{rel-respects-binary-pred} \{(a, b). (b, a) \in \text{Rel}\} \) Pred

by \text{blast}

ultimately

show \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \wedge \text{rel-respects-binary-pred Rel Pred} \)

qed

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

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

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

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

using \( \text{enc-preserves-pred-iff-indRelR-preserves-pred}[\text{where Pred=Pred}] \)

indRelR-preserves-pred-iff-indRelL-reflects-pred\[\text{where Pred=Pred}\]

by \text{blast}

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

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

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

using \( \text{enc-reflects-pred-iff-indRelR-reflects-pred}[\text{where Pred=Pred}] \)

indRelR-reflects-pred-iff-indRelL-preserves-pred\[\text{where Pred=Pred}\]

by \text{blast}

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

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

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

using \( \text{enc-respects-pred-iff-indRelR-reflects-pred}[\text{where Pred=Pred}] \)

indRelR-reflects-pred-iff-indRelL-preserves-pred\[\text{where Pred=Pred}\]

by \text{blast}

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

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

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

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

using \( \text{enc-preserves-pred-iff-source-target-rel-preserves-pred}[\text{where Pred=Pred}] \)

indRelR-cond-preservation-iff-indRelL-cond-reflection\[\text{where Pred=Pred}\]

by \text{simp}

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

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

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

using \( \text{enc-reflects-pred-iff-source-target-rel-reflects-pred}[\text{where Pred=Pred}] \)

indRelR-cond-reflection-iff-indRelL-cond-preservation\[\text{where Pred=Pred}\]

by \text{simp}

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

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

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

using \( \text{enc-respects-pred-iff-source-target-rel-respects-pred-encR}[\text{where Pred=Pred}] \)

indRelR-cond-respection-iff-indRelL-cond-respection\[\text{where Pred=Pred}\]

by \text{simp}
To analyse the respect 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.

\textbf{inductive-set (in encoding) indRel} \\
\quad :: \left((( \texttt{`procS}, `procT) \texttt{Proc}) \times ((`procS, `procT) \texttt{Proc}) \right) \texttt{set} \\
\quad \textbf{where} \\
\quad \texttt{encR: (SourceTerm S, TargetTerm ([S]))} \in \texttt{indRel} | \\
\quad \texttt{encL: (TargetTerm ([S]), SourceTerm S)} \in \texttt{indRel} \\

\textbf{abbreviation (in encoding) indRelInfix} :: \\
\quad (`procS, `procT) \texttt{Proc} \Rightarrow (procS, `procT) \texttt{Proc} \Rightarrow \texttt{bool} (~ \mathcal{R}[-] - [75, 75] 80) \\
\quad \textbf{where} \\
\quad P \equiv (P, Q) \in \texttt{indRel} \\

\textbf{lemma (in encoding) indRel-symm}:
\begin{align*}
\text{shows } \texttt{sym indRel} \\
\text{unfolding } \texttt{sym-def} \\
\text{by (auto simp add: indRel.simps indRel.encR indRel.encL)}
\end{align*}

\textbf{inductive-set (in encoding) indRelEQ} \\
\quad :: \left((( \texttt{`procS}, `procT) \texttt{Proc}) \times ((`procS, `procT) \texttt{Proc}) \right) \texttt{set} \\
\quad \textbf{where} \\
\quad \texttt{encR: (SourceTerm S, TargetTerm ([S]))} \in \texttt{indRelEQ} | \\
\quad \texttt{encL: (TargetTerm ([S]), SourceTerm S)} \in \texttt{indRelEQ} | \\
\quad \texttt{target: (TargetTerm T, TargetTerm T)} \in \texttt{indRelEQ} | \\
\quad \texttt{trans: [(P, Q) \in \texttt{indRelEQ}; (Q, R) \in \texttt{indRelEQ}] \Rightarrow (P, R) \in \texttt{indRelEQ}} \\

\textbf{abbreviation (in encoding) indRelEQInfix} :: \\
\quad (`procS, `procT) \texttt{Proc} \Rightarrow (procS, `procT) \texttt{Proc} \Rightarrow \texttt{bool} (~ \sim [-] - [75, 75] 80) \\
\quad \textbf{where} \\
\quad P \equiv (P, Q) \in \texttt{indRelEQ} \\

\textbf{lemma (in encoding) indRelEQ-refl}:
\begin{align*}
\text{shows } \texttt{refl indRelEQ} \\
\text{unfolding } \texttt{refl-on-def} \\
\text{proof auto} \\
\text{fix } P \\
\text{show } P \sim [-] P \\
\text{proof (cases P)} \\
\text{case (SourceTerm SP)} \\
\text{assume } SP \in S P \\
\text{moreover have } \texttt{SourceTerm SP} \sim [-] \texttt{TargetTerm ([SP])} \\
\text{by (rule indRelEQ.encR)} \\
\text{moreover have } \texttt{TargetTerm ([SP])} \sim [-] \texttt{SourceTerm SP} \\
\text{by (rule indRelEQ.encL)} \\
\text{ultimately show } P \sim [-] P \\
\text{by (simp add: indRelEQ.trans | where P=SourceTerm SP and Q=TargetTerm ([SP])}) \\
\text{next} \\
\text{case (TargetTerm TP)} \\
\text{assume } TP \in T P \\
\text{thus } P \sim [-] P \\
\text{by (simp add: indRelEQ.target)} \\
\text{qed} \\
\text{qed}
\end{align*}

\textbf{lemma (in encoding) indRelEQ-is-preorder}:
\begin{align*}
\text{shows preorder indRelEQ} \\
\text{unfolding preorder-on-def} \\
\text{proof} \\
\text{show refl indRelEQ} \\
\text{by (rule indRelEQ-refl)}
\end{align*}
next
show \textit{trans indRelEQ}
proof \textit{clarify}
fix \( P \), \( Q \), and \( R \)
assume \( P \sim \cdot \cdot \cdot Q \) and \( Q \sim \cdot \cdot \cdot R \)
thus \( P \sim \cdot \cdot \cdot R \)
by (rule indRelEQ.trans)
qed
qed

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

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

lemma (in encoding) \textit{refl-trans-closure-of-indRel}:
shows \textit{indRelEQ = indRel*}
proof \textit{auto}
fix \( P \), \( Q \)
assume \( P \sim \cdot \cdot \cdot Q \)
thus \( (P, Q) \in \text{indRel*} \)
proof \textit{induct}
case \textit{(encR \( S \))}
show \( (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRel*} \)
using \textit{indRel.encR[of \( S \)]}
by simp
next
case \textit{(encL \( S \))}
show \( (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{indRel*} \)
using \textit{indRel.encL[of \( S \)]}
by simp
next
72
case (target T)
  show (TargetTerm T, TargetTerm T) ∈ indRel*
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ indRel* and (Q, R) ∈ indRel*
  thus (P, R) ∈ indRel* 
    by simp
qed
next
  fix P Q
  assume (P, Q) ∈ indRel*
  thus P ∼ [
    by simp
  qed

lemma (in encoding) refl-symm-trans-closure-of-indRel:
  shows indRelEQ = (symcl (indRel−))∗
proof – 
  have (symcl (indRel−))∗ = (symcl indRel)∗
    by (rule refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRel])
  moreover have symcl indRel = indRel
    by (simp add: symcl-def indRelR symm-closure-of-symm-rel[where Rel=indRel])
  ultimately show indRelEQ = (symcl (indRel−))∗
    by (simp add: refl-trans-closure-of-indRel)
qed

lemma (in encoding) symm-closure-of-indRelR:
  shows indRel = symcl indRelR 
  and indRelEQ = (symcl (indRelR−))∗
proof – 
  show indRel = symcl indRelR 
  proof auto 
    fix P Q 
    assume P R = Q 
    thus (P, Q) ∈ symcl indRelR
      by (induct, simp-all add: symcl-def indRelR_encR)
  next 
    fix P Q 
    assume (P, Q) ∈ symcl indRelR 
    thus Q R = P 
      by (auto simp add: symcl-def indRelR_encR symps indRel_encR encL encR encL)
  qed 
  thus indRelEQ = (symcl (indRelR−))∗
    using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelR]
      refl-trans-closure-of-indRel
      by simp
qed
lemma (in encoding) symm-closure-of-indRelL:
  shows \( \text{indRel} = \text{symcl indRelL} \)
  and \( \text{indRelEQ} = (\text{symcl (indRelL\textsuperscript{\textsuperscript{\textdagger}})})^\dagger \)
proof
  show \( \text{indRel} = \text{symcl indRelL} \)
  proof auto
    fix \( P \ Q \)
    assume \( P \ R [\cdot] Q \)
    thus \( (P, Q) \in \text{symcl indRelL} \)
      by (induct, simp-all add: symcl-def indRelL.encL)
  next
    fix \( P \ Q \)
    assume \( (P, Q) \in \text{symcl indRelL} \)
    thus \( P \ R [\cdot] Q \)
      by (auto simp add: symcl-def indRelL.simps indRel.encR indRel.encL)
  qed
  thus \( \text{indRelEQ} = (\text{symcl (indRelL\textsuperscript{\textsuperscript{\textdagger}})})^\dagger \)
  using refl-symm-trans-closure-is-symm-refl-trans-closure
  where \( \text{Rel} = \text{indRelL} \)
  by simp
qed

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

lemma (in encoding) indRel-respects-pred-iff-indRelR-respects-pred:
  fixes \( \text{Pred} :: (\text{'procS}, \text{'procT}) \text{Proc} \Rightarrow \text{bool} \)
  shows \( \text{rel-respects-pred indRel \text{Pred} = rel-respects-pred indRelR \text{Pred} } \)
proof
  assume respection: \( \text{rel-respects-pred indRel \text{Pred} } \)
  show \( \text{rel-respects-pred indRelR \text{Pred} } \)
  proof auto
    fix \( P \ Q \)
    assume \( P \ R [\cdot] R Q \)
    from this obtain \( S \) where \( S \in S P \) and \( [S] \in T Q \)
      by (induct, blast)
    hence \( P \ R [\cdot] Q \)
      by (simp add: indRel.encR)
    moreover assume \( \text{Pred} P \)
    ultimately show \( \text{Pred} Q \)
      using respection
      by blast
  next
    fix \( P \ Q \)
    assume \( P \ R [\cdot] R Q \)
    from this obtain \( S \) where \( S \in S P \) and \( [S] \in T Q \)
      by (induct, blast)
    hence \( P \ R [\cdot] Q \)
      by (simp add: indRel.encR)
    moreover assume \( \text{Pred} Q \)
    ultimately show \( \text{Pred} P \)
      using respection
      by blast
  qed

next
  assume \( \text{rel-respects-pred indRelR \text{Pred} } \)
  thus \( \text{rel-respects-pred indRel \text{Pred} } \)
    using symm-closure-of-indRelR(1)
    respection-and-closures(2)[where \( \text{Rel} = \text{indRelR} \) and \( \text{Pred} = \text{Pred} \)]
    by blast
lemma (in encoding) indRel-respects-binary-pred-iff-indRelR-respects-binary-pred:
  fixes Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
  shows rel-respects-binary-pred indRel Pred = rel-respects-binary-pred indRelR Pred
proof
  assume respection: rel-respects-binary-pred indRel Pred
  show rel-respects-binary-pred indRelR Pred
  proof auto
    fix P Q x
    assume P R [\[ \cdot \\] \T Q
    from this obtain S where S ∈S P and [S] ∈T Q
    by (induct, blast)
    hence P R [\[ \cdot \\] \Q
    by (simp add: indRel.encR)
    moreover assume Pred P x
    ultimately show Pred Q x
    using respection
    by blast
  next
    fix P Q x
    assume P R [\[ \cdot \\] \R Q
    from this obtain S where S ∈S P and [S] ∈T Q
    by (induct, blast)
    hence P R [\[ \cdot \\] \Q
    by (simp add: indRel.encR)
    moreover assume Pred P x
    ultimately show Pred Q x
    using respection
    by blast
  qed
next
  assume rel-respects-binary-pred indRelR Pred
  thus rel-respects-binary-pred indRel Pred
  using symm-closure-of-indRelR(1)
  respection-of-binary-predicates-and-closures(2)[where Rel=indRelR and Pred=Pred]
  by blast
qed

lemma (in encoding) indRel-cond-respection-iff-indRelR-cond-respection:
  fixes Pred :: ('procS, 'procT) Proc ⇒ bool
  shows (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
        ∧ rel-respects-pred Rel Pred)
        = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel Pred)
proof
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel) ∧ rel-respects-pred Rel Pred
  from this obtain Rel
  where ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel
  and rel-respects-pred Rel Pred
  by blast
  thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel Pred
  by blast
next
  assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel Pred
  from this obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  and A2: rel-respects-pred Rel Pred
  by blast
  from A1 have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ symcl Rel
  ∧ (TargetTerm ([S]), SourceTerm S) ∈ symcl Rel
  by blast
by (simp add: symcl-def)

moreover from A2 have rel-respects-pred (symcl Rel) Pred
  using respection-and-closures(2)[where Rel=Rel and Pred=Pred]
  by blast

ultimately

show ∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
  ∧ rel-respects-pred Rel Pred
  by blast

qed

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

  fixes Pred :: ('procS, 'procT) Proc ⇒ bool

  shows (∃Rel.
    (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
    ∧ rel-respects-binary-pred Rel Pred
    ⇒ (∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
        ∧ rel-respects-binary-pred Rel Pred)

proof

  assume ∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel ∧ rel-respects-binary-pred Rel Pred

  from this obtain Rel

  where ∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel
  and rel-respects-binary-pred Rel Pred
  by blast

  thus ∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ rel-respects-binary-pred Rel Pred
  by blast

next

  assume ∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ rel-respects-binary-pred Rel Pred

  from this obtain Rel where A1: ∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
    and A2: rel-respects-binary-pred Rel Pred
  by blast

  from A1 have ∀S. (SourceTerm S, TargetTerm ([S])) ∈ symcl Rel
    ∧ (TargetTerm ([S]), SourceTerm S) ∈ symcl Rel
  by (simp add: symcl-def)

  moreover from A2 have rel-respects-binary-pred (symcl Rel) Pred

  using respection-of-binary-predicates-and-closures(2)[where Rel=Rel and Pred=Pred]
  by blast

  ultimately

  show ∃Rel. (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
    ∧ rel-respects-binary-pred Rel Pred
  by blast

qed

An encoding respects a predicate iff indRel respects this predicate.

lemma (in encoding) enc-respects-pred-iff-indRel-respects-pred:

  fixes Pred :: ('procS, 'procT) Proc ⇒ bool

  shows enc-respects-pred Pred = rel-respects-pred indRel Pred

  using enc-respects-pred-iff-indRelR-respects-pred[where Pred=Pred]
  by simp

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

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

  fixes Pred :: ('procS, 'procT) Proc ⇒ bool

  shows enc-respects-pred Pred
     = (∃Rel.
         (∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
         ∧ rel-respects-pred Rel Pred)

  using enc-respects-pred-iff-source-target-rel-respects-pred-encRL[where Pred=Pred]

  by (simp add: symcl-def)
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) inducRelR**

\[ (\text{'procT} \times \text{'procT}) \subseteq ((\text{'procS}, \text{'procT}) \text{ Proc}) \times ((\text{'procS}, \text{'procT}) \text{ Proc}) \]

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

**where**

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

**target: (T1, T2) \in TRel \Longrightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{indRelRT} \text{ TRel} **

**abbreviation (in encoding) inducRelRTR**

\[ (\text{'procS}, \text{'procT}) \text{ Proc} \Rightarrow (\text{'procT} \times \text{'procT}) \subseteq ((\text{'procS}, \text{'procT}) \text{ Proc}) \Rightarrow \text{bool} \]

\[-R[\text{'procT}] \text{ RT} < - [75, 75, 75, 80] \]

**where**

\[ P \ R[\text{'procT}] \text{ RT} < TRel > Q \equiv (P, Q) \in \text{indRelRT} \text{ TRel} **

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

\[ (\text{'procT} \times \text{'procT}) \subseteq ((\text{'procS}, \text{'procT}) \text{ Proc}) \times ((\text{'procS}, \text{'procT}) \text{ Proc}) \]

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

**where**

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

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

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

**trans: [[P, Q] \in \text{indRelRTPO} \text{ Trel}; (Q, R) \in \text{indRelRTPO} \text{ TRel}] \Longrightarrow (P, R) \in \text{indRelRTPO} \text{ TRel} **

**abbreviation (in encoding) inducRelRTPOref**

\[ (\text{'procS}, \text{'procT}) \text{ Proc} \Rightarrow (\text{'procT} \times \text{'procT}) \subseteq ((\text{'procS}, \text{'procT}) \text{ Proc}) \Rightarrow \text{bool} \]

\[-\text{'procT} \text{ RT} < - [75, 75, 75, 80] \]

**where**

\[ P \text{'procT} \text{ RT} < TRel > Q \equiv (P, Q) \in \text{indRelRTPO} \text{ TRel} **

**lemma (in encoding) inducRelRTPO-refl**

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

**assumes** refl:: refl TRel

**shows** refl (indRelRTPO TRel)

**unfolding** refl-on-def

**proof auto**

**fix P**

**show** P \text{'procT} \text{ RT} < TRel > P

**proof (cases P)**

**case** (SourceTerm SP)

**assume** SP \in S P

**thus** P \text{'procT} \text{ RT} < TRel > P

**by (simp add: inducRelRTPO.source)**

**next**

**case** (TargetTerm TP)

**assume** TP \in T P

**with** refl **show** P \text{'procT} \text{ RT} < TRel > P

**unfolding** refl-on-def

**by (simp add: inducRelRTPO.target)**

**qed**

**qed**

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

**fixes** TRel :: (\text{'procT} \times \text{'procT}) set
assumes refl :: refl TRel
shows indRelRTPO TRel = (indRelRT TRel)^*
proof auto
  fix P Q
  assume P ≤_RT< TRel> Q
  thus (P, Q) ∈ (indRelRT TRel)^*
proof induct
  case (encR S)
  show (SourceTerm S, TargetTerm [S]) ∈ (indRelRT TRel)^*
    using indRelRT.encR[of S TRel]
    by simp
next
  case (source S)
  show (SourceTerm S, SourceTerm S) ∈ (indRelRT TRel)^*
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  thus (TargetTerm T1, TargetTerm T2) ∈ (indRelRT TRel)^*
    using indRelRT.target[of T1 T2 TRel]
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ (indRelRT TRel)^* and (Q, R) ∈ (indRelRT TRel)^*
  thus (P, R) ∈ (indRelRT TRel)^*
    by simp
qed
next
  fix P Q
  assume (P, Q) ∈ (indRelRT TRel)^*
  thus P ≤_RT< TRel> Q
proof induct
  from refl show P ≤_RT< TRel> P
    using indRelRTPO-refl[of TRel]
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume P ≤_RT< TRel> Q
  moreover assume Q ≤_RT< TRel> R
  hence Q ≤_RT< TRel> R
    by (induct, simp-all add: indRelRTPO.encR indRelRTPO.target)
  ultimately show P ≤_RT< TRel> R
    by (rule indRelRTPO.trans)
qed

lemma (in encoding) indRelRTPO-is-preorder:
  fixes TRel :: '(procT × 'procT) set
  assumes refl :: refl TRel
  shows preorder (indRelRTPO TRel)
  unfolding preorder-on-def
proof
  from refl show refl (indRelRTPO TRel)
    by (rule indRelRTPO-refl)
next
  show trans (indRelRTPO TRel)
    unfolding trans-def
proof clarify
  fix P Q R
  assume P ≤_RT< TRel> Q and Q ≤_RT< TRel> R
lemma (in encoding) transitive-closure-of-TRel-to-indRelRTPO:
  fixes TRel :: ('procT × 'procT) set
  and TP TQ :: 'procT
  shows (TP, TQ) ∈ TRel⁺ → TargetTerm TP ≤ [+]RT < TRel> TargetTerm TQ
proof
  assume (TP, TQ) ∈ TRel⁺
  thus TargetTerm TP ≤ [+]RT < TRel> TargetTerm TQ
by (rule indRelRTPO.target)
next
case (step TQ TR)
  assume TargetTerm TP ≤ [+]RT < TRel> TargetTerm TQ
  moreover assume (TQ, TR) ∈ TRel
  hence TargetTerm TQ ≤ [+]RT < TRel> TargetTerm TR
  by (simp add: indRelRTPO.target)
  ultimately show TargetTerm TP ≤ [+]RT < TRel> TargetTerm TR
  by (rule indRelRTPO.trans)
qed
thus $\forall (P, Q) \in \text{indRelR}. \text{Pred} (P, Q)$

by (auto simp add: indRelR.simps)

qed

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

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

shows $(\forall TRel. (\forall (TP, TQ) \in TRel. \text{Pred} (\text{TargetTerm} \ TP, \text{TargetTerm} \ TQ))
\leftrightarrow (\forall (P, Q) \in \text{indRelRT} \ TRel. \text{Pred} (P, Q)))$

= $(\exists \text{Rel}. (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([T])) \in \text{Rel}) \land (\forall (P, Q) \in \text{Rel}. \text{Pred} (P, Q)))$

using indRelRT-iff-exists-source-target-relation [where $\text{Pred} = \text{Pred}$]

indRelR-modulo-pred-impl-indRelRT-modulo-pred [where $\text{Pred} = \text{Pred}$]

by simp

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

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

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

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

and transCond: $\forall P Q R. \text{Pred} (P, Q) \land \text{Pred} (Q, R) \rightarrow \text{Pred} (P, R)$

shows $(\forall (P, Q) \in \text{indRelRT} \text{TRel}. \text{Pred} (P, Q)) = (\forall (P, Q) \in \text{indRelRTPO} \text{TRel}. \text{Pred} (P, Q))$

proof auto

fix $P Q$

assume $A: \forall x \in \text{indRelRT} \text{TRel}. \text{Pred} \ x$

assume $P \leq [(\text{'}\text{RT} < \text{TRel} > Q$

thus $\text{Pred} (P, Q)$

proof induct

case (encR $S$)

have $\text{SourceTerm} \ S \ R[(\text{'}\text{RT} < \text{TRel} > \text{TargetTerm} \ ([T]))$

by (simp add: indRelRT.encR)

with $A$ show $\text{Pred} (\text{SourceTerm} \ S, \text{SourceTerm} \ S)$

by simp

next

case (source $S$)

from reflCond show $\text{Pred} (\text{SourceTerm} \ S, \text{SourceTerm} \ S)$

by simp

next

case (target $T1 T2$)

assume $(T1, T2) \in \text{TRel}$

hence $\text{TargetTerm} \ T1 \ R[(\text{'}\text{RT} < \text{TRel} > \text{TargetTerm} \ T2$

by (simp add: indRelRT.target)

with $A$ show $\text{Pred} (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2)$

by simp

next

case (trans $P Q R$)

assume $\text{Pred} (P, Q)$ and $\text{Pred} (Q, R)$

with transCond show $\text{Pred} (P, R)$

by blast

qed

next

fix $P Q$

assume $\forall x \in \text{indRelRTPO} \text{TRel}. \text{Pred} \ x$ and $P \leq [(\text{'}\text{RT} < \text{TRel} > Q$

thus $\text{Pred} (P, Q)$

by (auto simp add: indRelRTPO.encR indRelRTPO.target indRelRT.simps)

qed

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

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

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

and $\forall P Q R. \text{Pred} (P, Q) \land \text{Pred} (Q, R) \rightarrow \text{Pred} (P, R)$

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

= $(\forall \text{TRel}. (\forall (TP, TQ) \in \text{TRel.} \text{Pred} (\text{TargetTerm} \ TP, \text{TargetTerm} \ TQ))$

$\leftrightarrow (\forall (P, Q) \in \text{indRelRTPO} \text{TRel.} \text{Pred} (P, Q)))$
The relation \( \text{indRelLT} \) includes \( \text{TRel} \) and relates literal translations and their source terms.

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

\[
\begin{align*}
\text{indRelLT} &:: (\text{procT} \times \text{procT}) \rightarrow (((\text{procS}, \text{'procT}) \text{Proc}) \times ((\text{procS}, \text{'procT}) \text{Proc}) \rightarrow \text{bool} \\
\text{for TRel} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{where} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\end{align*}
\]

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

\[
\begin{align*}
\text{indRelLTinfix} &:: (\text{procT} \times \text{procT}) \rightarrow (((\text{procS}, \text{'procT}) \text{Proc}) \times ((\text{procS}, \text{'procT}) \text{Proc}) \rightarrow \text{bool} \\
\text{for TRel} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{where} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\end{align*}
\]

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

\[
\begin{align*}
\text{indRelLTO} &:: (\text{procT} \times \text{procT}) \rightarrow (((\text{procS}, \text{'procT}) \text{Proc}) \times ((\text{procS}, \text{'procT}) \text{Proc}) \rightarrow \text{bool} \\
\text{for TRel} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{where} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\end{align*}
\]

**abbreviation (in encoding) \( \text{indRelLTOinfix} \)**

\[
\begin{align*}
\text{indRelLTOinfix} &:: (\text{procT} \times \text{procT}) \rightarrow (((\text{procS}, \text{'procT}) \text{Proc}) \times ((\text{procS}, \text{'procT}) \text{Proc}) \rightarrow \text{bool} \\
\text{for TRel} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{where} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\end{align*}
\]

**lemma (in encoding) \( \text{indRelLTO-refl} \)**

\[
\begin{align*}
\text{fixes} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{assumes} &:: \text{refl} \rightarrow \text{TRel} \\
\text{shows} &:: \text{refl} \rightarrow \text{TRel} \\
\text{unfolding} &:: \text{refl-on-def} \\
\text{proof} &:: \text{auto} \\
\text{show} &:: \text{P} \rightarrow \text{P} \\
\text{proof} (\text{cases} \text{P}) \\
\text{case} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{case} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{thus} &:: \text{P} \rightarrow \text{P} \\
\text{by} &:: \text{simp add: indRelLTO.source} \\
\end{align*}
\]

next

\[
\begin{align*}
\text{case} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{assumption} &:: \text{P} \rightarrow \text{P} \\
\end{align*}
\]

**proof auto**

\[
\begin{align*}
\text{fix} &:: \text{P} \\
\text{show} &:: \text{P} \rightarrow \text{P} \\
\text{proof} (\text{cases} \text{P}) \\
\text{case} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{assume} &:: \text{P} \rightarrow \text{P} \\
\text{thus} &:: \text{P} \rightarrow \text{P} \\
\text{by} &:: \text{simp add: indRelLTO.source} \\
\text{next} &:: (\text{procT} \times \text{procT}) \rightarrow \text{bool} \\
\text{assumption} &:: \text{P} \rightarrow \text{P} \\
\text{with} &:: \text{refl} \rightarrow \text{P} \rightarrow \text{P} \\
\text{using} &:: \text{indRelLTO.target} \rightarrow \text{TRel} \rightarrow \text{TRel} \\
\text{unfolding} &:: \text{refl-on-def} \\
\end{align*}
\]
lemma (in encoding) refl-trans-closure-of-indRelLT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows indRelLTPO TRel = (indRelLT TRel)*
proof auto
  fix P Q
  assume P ≲[
      LT < TRel>
    ] Q
  thus (P, Q) ∈ (indRelLT TRel)*
proof induct
    case (encL S)
    show (TargetTerm ([S]), SourceTerm S) ∈ (indRelLT TRel)*
      using indRelLT.encL[of S TRel]
      by simp
  next
    case (source S)
    show (SourceTerm S, SourceTerm S) ∈ (indRelLT TRel)*
      by simp
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    thus (TargetTerm T1, TargetTerm T2) ∈ (indRelLT TRel)*
      using indRelLT.target[of T1 T2 TRel]
      by simp
  next
    case (trans P Q R)
    assume (P, Q) ∈ (indRelLT TRel)* and (Q, R) ∈ (indRelLT TRel)*
    thus (P, R) ∈ (indRelLT TRel)*
      by simp
qed

next
  fix P Q
  assume (P, Q) ∈ (indRelLT TRel)*
  thus P ≲[
      LT < TRel>
    ] Q
proof induct
  from refl show P ≲[
      LT < TRel>
    ] P
  using indRelLTPO-refl[of TRel]
  unfolding refl-on-def
  by simp
next
  case (step Q R)
  assume P ≲[
      LT < TRel>
    ] Q
  moreover assume Q R ≲[
      LT < TRRel>
    ] R
  hence Q ≲[
      LT < TRrel>
    ] R
  by (induct, simp-all add: indRelLTPO.encL indRelLTPO.target)
  ultimately show P ≲[
      LT < TRel>
    ] R
  by (rule indRelLTPO.trans)
qed

inductive-set (in encoding) indRelT
  :: ('procT × 'procT) set ⇒ (((['procS', 'procT) Proc) × (['procS', 'procT) Proc)) set
for TRel :: ('procT × 'procT) set
where
  encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelT TRel |
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelT TRel |
target: (T1, T2) ∈ TRel ⇒ (TargetTerm T1, TargetTerm T2) ∈ indRelT TRel
abbrev (in encoding) indRelTInfix
  :: ('procS, 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
  (- R[\cdot]T<> - [75, 75, 75] 80)
where
  P \mathcal{R}[\cdot]T\mathcal{R}\mathcal{L} Q \equiv (P, Q) ∈ indRelT TRel

lemma (in encoding) indRelT-symm:
  fixes TRel :: ('procT × 'procT) set
  assumes symm: sym TRel
  shows sym (indRelT TRel)
unfolding sym-def
proof clarify
  fix P Q
  assume (P, Q) ∈ indRelT TRel
  thus (Q, P) ∈ indRelT TRel
    using symm
    unfolding sym-def
    by (induct, simp-all add: indRelT.\texttt{encL} indRelT.\texttt{encL} indRelT.\texttt{target})
qed

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

abbreviation (in encoding) indRelTEQinfix
  :: ('procT × 'procT) Proc ⇒ ('procT × 'procT) set ⇒ ('procS, 'procT) Proc ⇒ bool
  (- \cdot<\cdot>- [75, 75, 75] 80)
where
  P \cdot<\cdot>T\cdot<\cdot> Q \equiv (P, Q) ∈ indRelTEQ TRel

lemma (in encoding) indRelTEQ-refl:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows refl (indRelTEQ TRel)
  unfolding refl-on-def
proof auto
  fix P
  show P \cdot<\cdot>T\cdot<\cdot> P
  proof (cases P)
    case (SourceTerm SP)
    assume SP ∈ S P
    moreover have SourceTerm SP \cdot<\cdot>T\cdot<\cdot> TargetTerm ([SP])
      by (rule indRelTEQ.\texttt{encR})
    moreover have TargetTerm ([SP]) \cdot<\cdot>T\cdot<\cdot> SourceTerm SP
      by (rule indRelTEQ.\texttt{encL})
    ultimately show P \cdot<\cdot>T\cdot<\cdot> P
      by (simp add: indRelTEQ.trans[where P=SourceTerm SP and Q=TargetTerm ([SP])])
  next
    case (TargetTerm TP)
    assume TP ∈ T P
    with refl show P \cdot<\cdot>T\cdot<\cdot> P
      unfolding refl-on-def
      by (simp add: indRelTEQ.\texttt{target})
  qed
  qed
lemma (in encoding) indRelTEQ-symm:
  fixes TRel :: ('procT × 'procT) set
  assumes symm: sym TRel
  shows sym (indRelTEQ TRel)
  unfolding sym-def
proof clarify
  fix P Q
  assume P ∼[[ ]]T<TRel> Q
  thus Q ∼[[ ]]T<TRel> P
proof induct
  case (encR S)
  show TargetTerm ([S]) ∼[[ ]]T<TRel> SourceTerm S
    by (rule indRelTEQ.encL(S TRel))
next
  case (encL S)
  show SourceTerm S ∼[[ ]]T<TRel> TargetTerm ([S])
    by (rule indRelTEQ.encR(S TRel))
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  with symm show TargetTerm T2 ∼[[ ]]T<TRel> TargetTerm T1
    unfolding sym-def
    by (simp add: indRelTEQ.target(S TRel))
next
  case (trans P Q R)
  assume R ∼[[ ]]T<TRel> Q and Q ∼[[ ]]T<TRel> P
  thus R ∼[[ ]]T<TRel> P
  by (rule indRelTEQ.trans(S TRel))
qed

lemma (in encoding) refl-trans-closure-of-indRelT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
  shows indRelTEQ TRel = (indRelT TRel)*
proof auto
  fix P Q
  assume P ∼[[ ]]T<TRel> Q
  thus (P, Q) ∈ (indRelT TRel)*
proof induct
  case (encR S)
  show (SourceTerm S, TargetTerm ([S])) ∈ (indRelT TRel)*
    using indRelT.encR(S TRel)
    by simp
next
  case (encL S)
  show (TargetTerm ([S]), SourceTerm S) ∈ (indRelT TRel)*
    using indRelT.encL(S TRel)
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  thus (TargetTerm T1, TargetTerm T2) ∈ (indRelT TRel)*
    using indRelT.target(S TRel)
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ (indRelT TRel)* and (Q, R) ∈ (indRelT TRel)*
  thus (P, R) ∈ (indRelT TRel)*
    by simp
qed
next
  fix P Q
  assume \((P, Q) \in (\text{indRelT \ TRel})^+\)
  thus \(P \symcl \mathbin{\langle \cdot \rangle} Q\)
proof
  induct
  from refl show \(P \symcl \mathbin{\langle \cdot \rangle} P\)
    using indRelTEQ-refl[of TRel]
  unfolding refl-on-def
  by simp
next
  case (step Q R)
  assume \(P \symcl \mathbin{\langle \cdot \rangle} Q\)
moreover assume \(Q \symcl \mathbin{\langle \cdot \rangle} R\)
  hence \(P \symcl \mathbin{\langle \cdot \rangle} R\)
    by (induct, simp-all add: indRelTEQ.encR indRelTEQ.encL indRelTEQ.target)
ultimately show \(P \symcl \mathbin{\langle \cdot \rangle} R\)
by (rule indRelTEQ.trans)
qed

lemma (in encoding) refl-symm-trans-closure-of-indRelT:
  fixes TRel :: ('procT \times 'procT) set
  assumes refl: \(\text{refl \ TRel}\)
    and symm: \(\text{sym \ TRel}\)
  shows \(\text{indRelTEQ \ TRel} = (\text{symcl ((indRelT \ TRel)})^+\)\)
proof
  have \((\text{symcl ((indRelT \ TRel)})^+\) = \((\text{symcl (indRelT \ TRel)})^+\)
    by (rule refl-symm-trans-closure-is-symm-refl-trans-closure[where \(\text{Rel} = \text{indRelT \ TRel}\)])
moreover from symm have \(\text{symcl \ (indRelT \ TRel)} = \text{indRelT \ TRel}\)
  using indRelT-symm[where \(\text{TRel} = \text{TRel} \text{symm-closure-of-symm-rel}[where \(\text{Rel} = \text{indRelT \ TRel}\)]
by blast
ultimately show \(\text{indRelTEQ \ TRel} = (\text{symcl ((indRelT \ TRel)})^+\)
  using refl refl-trans-closure-of-indRelT[where \(\text{TRel} = \text{TRel}\]
by simp
qed

lemma (in encoding) symm-closure-of-indRelRT:
  fixes TRel :: ('procT \times 'procT) set
  assumes refl: \(\text{refl \ TRel}\)
    and symm: \(\text{sym \ TRel}\)
  shows \(\text{indRelRT \ TRel} = \text{symcl \ (indRelRT \ TRel)}\)
    and \(\text{indRelTEQ \ TRel} = (\text{symcl ((indRelRT \ TRel)})^+\)\)
proof
  show \(\text{indRelRT \ TRel} = \text{symcl \ (indRelRT \ TRel)}\)
proof auto
  fix P Q
  assume \(P \symcl \mathbin{\langle \cdot \rangle} Q\)
  thus \((P, Q) \in \text{symcl \ (indRelRT \ TRel)}\)
    by (induct, simp-all add: symcl-def indRelRT.encR indRelRT.target)
next
  fix P Q
  assume \((P, Q) \in \text{symcl \ (indRelRT \ TRel)}\)
  thus \(P \symcl \mathbin{\langle \cdot \rangle} Q\)
proof (auto simp add: symcl-def indRelRT.simps)
  fix S
  show \(\text{SourceTerm \ S \symcl \mathbin{\langle \cdot \rangle} \text{TargetTerm \ ([S])}}\)
    by (rule indRelT.encR)
next
  fix T1 T2
  assume \((T1, T2) \in \text{TRel}\)
  thus \(\text{TargetTerm \ T1 \symcl \mathbin{\langle \cdot \rangle} \text{TargetTerm \ T2}}\)
by (rule indRelT.target)

next
fix S
show TargetTerm ([S]) R[·] T< TRel> SourceTerm S
by (rule indRelT.encL)

next
fix T1 T2
assume (T1, T2) ∈ TRel
with symm show TargetTerm T2 R[·] T< TRel> TargetTerm T1
  unfolding sym-def
by (simp add: indRelT.target)

qed

lemma (in encoding) symm-closure-of-indRelLT:
  fixes TRel :: ('procT × 'procT) set
  assumes refl: refl TRel
and symm: sym TRel
  shows indRelLT TRel = symcl (indRelLT TRel)
and indRelTEQ TRel = (symcl ((indRelLT TRel)−1))

proof —
show indRelLT TRel = symcl (indRelLT TRel)
proof auto
fix P Q
assume P R[·] T< TRel> Q
thus (P, Q) ∈ symcl (indRelLT TRel)
  by (induct, simp-all add: symcl-def indRelLT.encL indRelLT.target)
next
fix P Q
assume (P, Q) ∈ symcl (indRelLT TRel)
thus P R[·] T< TRel> Q
proof (auto simp add: symcl-def indRelLT.simps)
  show SourceTerm S R[·] T< TRel> TargetTerm ([S])
    by (rule indRelT.encR)
next
fix T1 T2
assume (T1, T2) ∈ TRel
thus TargetTerm T1 R[·] T< TRel> TargetTerm T2
  by (rule indRelT.target)
next
fix S
show TargetTerm ([S]) R[·] T< TRel> SourceTerm S
  by (rule indRelT.encL)
next
fix T1 T2
assume (T1, T2) ∈ TRel
with symm show TargetTerm T2 R[·] T< TRel> TargetTerm T1
  unfolding sym-def
by (simp add: indRelT.target)
qed

qed

with refl show indRelTEQ TRel = (symcl ((indRelLT TRel)−1))
using refl-symm-trans-closure-is-symm-refl-trans-closure[where Rel=indRelLT TRel]
  refl-trans-closure-of-indRelT
by simp
proof

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

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

```plaintext
fixes TRel :: (\'procT × \'procT) set
and TP TQ :: \'procT
assumes rel: TargetTerm TP \( R[^\text{RT}] < \text{TRel} > \) TargetTerm TQ
shows (TP, TQ) \( \in \text{TRel} \)
  using rel
  by (simp add: indRelRT.simps)
```

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

```plaintext
fixes TRel :: (\'procT × \'procT) set
and TP TQ :: \'procT
assumes rel: TargetTerm TP \( R[^\text{LT}] < \text{TRel} > \) TargetTerm TQ
shows (TP, TQ) \( \in \text{TRel} \)
  using rel
  by (simp add: indRelLT.simps)
```

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

```plaintext
fixes TRel :: (\'procT × \'procT) set
and TP TQ :: \'procT
assumes rel: TargetTerm TP \( R[^\text{T}] < \text{TRel} > \) TargetTerm TQ
shows (TP, TQ) \( \in \text{TRel} \)
  using rel
  by (simp add: indRelT.simps)
```

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

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

```plaintext
fixes TRel :: (\'procT × \'procT) set
and P Q :: \'procT
assumes rel: P \( \leq[^\text{RT}] \text{TRel} > \) Q
shows \( \forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow SP = SQ \)
    and \( \forall SP TQ. SP \in S P \land TQ \in T Q \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
    and \( \forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow \text{False} \)
    and \( \forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in \text{TRel}^+ \)
proof –
  have reflRel: \( \forall S. ([S], [S]) \in \text{TRel} \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\} \)
  by auto
  from rel show \( \forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow SP = SQ \)
    and \( \forall SP TQ. SP \in S P \land TQ \in T Q \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
    and \( \forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow \text{False} \)
    and \( \forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in \text{TRel}^+ \)
proof induct
  case (encR S)
  show \( \forall SP SQ. SP \in S \text{SourceTerm} S \land SQ \in S \text{TargetTerm} ([S]) \rightarrow SP = SQ \)
    and \( \forall TP SQ. TP \in T \text{SourceTerm} S \land SQ \in S \text{TargetTerm} ([S]) \rightarrow \text{False} \)
    and \( \forall TP TQ. TP \in T \text{SourceTerm} S \land TQ \in T \text{TargetTerm} ([S]) \rightarrow (TP, TQ) \in \text{TRel}^+ \)
    by simp-all
  from reflRel show \( \forall SP TQ. SP \in S \text{SourceTerm} S \land TQ \in T \text{TargetTerm} ([S]) \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
    by blast
  next
    case (source S)
    show \( \forall SP SQ. SP \in S \text{SourceTerm} S \land SQ \in S \text{SourceTerm} S \rightarrow SP = SQ \)
```

87
by simp

show \( \forall SP \ TQ. \ SP \in S \ \text{SourceTerm} \ S \land TQ \in T \ \text{SourceTerm} \ S \)
\( \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

and \( \forall TP \ TQ. \ TP \in T \ \text{SourceTerm} \ S \land SQ \in S \ \text{SourceTerm} \ S \rightarrow \text{False} \)

and \( \forall TP \ TQ. \ TP \in T \ \text{SourceTerm} \ T1 \land TQ \in T \ \text{SourceTerm} \ T2 \)
\( \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

by simp-all

next

case (target \( T1, T2 \))

show \( \forall SP \ SQ. \ SP \in S \ \text{TargetTerm} \ T1 \land SQ \in S \ \text{TargetTerm} \ T2 \rightarrow SP = SQ \)

and \( \forall SP \ TQ. \ SP \in S \ \text{TargetTerm} \ T1 \land SP \in T \ \text{TargetTerm} \ T2 \)
\( \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

by simp-all

assume \( (T1, T2) \in \text{TRel} \)

thus \( \forall TP \ TQ. \ TP \in T \ \text{TargetTerm} \ T1 \land TQ \in T \ \text{TargetTerm} \ T2 \rightarrow (TP, TQ) \in \text{TRel}^+ \)

by simp

next

case (trans \( P \ Q \ R \))

assume A1: \( \forall SP \ SQ. \ SP \in S \ P \land SQ \in S \ Q \rightarrow SP = SQ \)

and A2: \( \forall SP \ TQ. \ SP \in S \ P \land TQ \in T \ Q \)
\( \rightarrow ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

and A3: \( \forall TP \ SQ. \ TP \in T \ P \land SQ \in S \ Q \rightarrow \text{False} \)

and A4: \( \forall TP \ TQ. \ TP \in T \ P \land TQ \in T \ Q \rightarrow (TP, TQ) \in \text{TRel}^+ \)

and A5: \( \forall SQ SR. \ SQ \in S \ Q \land SR \in S \ R \rightarrow SQ = SR \)

and A6: \( \forall SQ TR. \ SQ \in S \ Q \land TR \in T \ R \)
\( \rightarrow ([SQ], TR) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

and A7: \( \forall TQ SR. \ TQ \in T \ Q \land SR \in S \ R \rightarrow \text{False} \)

and A8: \( \forall TQ TR. \ TQ \in T \ Q \land TR \in T \ R \rightarrow (TQ, TR) \in \text{TRel}^+ \)

show \( \forall SP SR. \ SP \in S \ P \land SR \in S \ R \rightarrow SP = SR \)

proof (cases \( Q \))

case (SourceTerm \( S \))

assume \( SQ \in S \ Q \)

with A1 A5 show \( \forall SP SR. \ SP \in S \ P \land SR \in S \ R \rightarrow SP = SR \)

by blast

next

case (TargetTerm \( TQ \))

assume \( TQ \in T \ Q \)

with A7 show \(?thesis\)

by blast

qed

show \( \forall SP TR. \ SP \in S \ P \land TR \in T \ R \)
\( \rightarrow ([SP], TR) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

proof (cases \( Q \))

case (SourceTerm \( S \))

assume \( SQ \in S \ Q \)

with A1 A6 show \(?thesis\)

by blast

next

case (TargetTerm \( TQ \))

assume A9: \( TQ \in T \ Q \)

show \( \forall SP TR. \ SP \in S \ P \land TR \in T \ R \)
\( \rightarrow ([SP], TR) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

proof clarify

fix \( SP \ TR \)

assume \( SP \in S \ P \)

with A2 A9 have \( ([SP], TQ) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)

by simp

moreover assume \( TR \in T \ R \)

with A8 A9 have \( (TQ, TR) \in \text{TRel}^+ \)

by simp

hence \( (TQ, TR) \in (\text{TRel} \cup \{(T1, T2), \exists S. \ T1 = [S] \land T2 = [S]\})^+ \)
proof induct
fix T2
assume (TQ, T2) ∈ TRel
thus (TQ, T2) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
by blast
next
case (step T2 T3)
assume (TQ, T2) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
moreover assume (T2, T3) ∈ TRel
hence (T2, T3) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
by blast
ultimately show (TQ, T3) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
by simp
qed
ultimately show ([SP], TR) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
by simp
qed
show ∀ TP SR. TP ∈ T P ∧ SR ∈ S R → False
proof (cases Q)
case (SourceTerm SQ)
assume SQ ∈ S Q
with A3 show ⊨ thesis
by blast
next
case (TargetTerm TQ)
assume TQ ∈ T Q
with A7 show ⊨ thesis
by blast
qed
show ∀ TP TR. TP ∈ T P ∧ TR ∈ T R → (TP, TR) ∈ TRel+
proof (cases Q)
case (SourceTerm SQ)
assume SQ ∈ S Q
with A3 show ⊨ thesis
by blast
next
case (TargetTerm TQ)
assume TQ ∈ T Q
with A4 A8 show ∀ TP TR. TP ∈ T P ∧ TR ∈ T R → (TP, TR) ∈ TRel+
by auto
qed
qed
qed

lemma (in encoding) indRelLTPO-to-TRel:
  fixes TRel :: ('procT × 'procT) set
  and P Q :: ('procS, 'procT) Proc
  assumes rel :: P ≤ A LT< TRel > Q
  shows ∀ SP SQ. SP ∈ S P ∧ SQ ∈ S Q → SP = SQ
  and ∀ SP TQ. SP ∈ S P ∧ TQ ∈ T Q → False
  and ∀ TP SQ. TP ∈ T P ∧ SQ ∈ S Q
                → (TP, [SQ]) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
  and ∀ TP TQ. TP ∈ T P ∧ TQ ∈ T Q → (TP, TQ) ∈ TRel+
proof
  have refTRel: ∀ S. ([S], [S]) ∈ TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]}
    by auto
  from rel show ∀ SP SQ. SP ∈ S P ∧ SQ ∈ S Q → SP = SQ
    and ∀ SP TQ. SP ∈ S P ∧ TQ ∈ T Q → False
    and ∀ TP SQ. TP ∈ T P ∧ SQ ∈ S Q
                → (TP, [SQ]) ∈ (TRel ∪ {(T1, T2). ∃ S. T1 = [S] ∧ T2 = [S]})+
    by blast
and \( \forall TP \ TQ. \ TP \in T \land TQ \in T \rightarrow (TP, \ TQ) \in TRel^+ \)

**proof** \textit{induct}

**case** (\textit{encL S})

\( \begin{aligned} & \text{show} \ \forall SP \ SQ. \ SP \in S \ SourceTerm \ (\llbracket S \rrbracket) \land SQ \in S \ SourceTerm \ S \rightarrow SP = SQ \\
& \land \forall SP \ TQ. \ SP \in S \ SourceTerm \ S \land TQ \in T \ SourceTerm \ S \rightarrow False \\
& \land \forall TP \ TQ. \ TP \in T \ SourceTerm \ (\llbracket S \rrbracket) \land TQ \in T \ SourceTerm \ S \rightarrow (TP, \ TQ) \in TRel^+ \\
& \text{by simp-all} \end{aligned} \)

\( \text{from reflTRel show} \ \forall TP \ SQ. \ TP \in T \ SourceTerm \ (\llbracket S \rrbracket) \land SQ \in S \ SourceTerm \ S \rightarrow (TP, \ [SQ]) \in (TRel \cup \{(T1, \ T2). \ \exists S. \ T1 = \llbracket S \rrbracket \land T2 = \llbracket S \rrbracket\})^+ \)

\( \text{by blast} \)

**next**

**case** (\textit{source S})

\( \begin{aligned} & \text{show} \ \forall SP \ SQ. \ SP \in S \ SourceTerm \ S \land SQ \in S \ SourceTerm \ S \rightarrow SP = SQ \\
& \text{by simp} \end{aligned} \)

\( \begin{aligned} & \text{show} \ \forall TP \ TQ. \ SP \in S \ SourceTerm \ S \land TQ \in T \ SourceTerm \ S \rightarrow False \\
& \text{by simp-all} \end{aligned} \)

**thus** \( \forall TP \ TQ. \ TP \in T \ SourceTerm \ T1 \land TQ \in T \ SourceTerm \ T2 \rightarrow (TP, \ TQ) \in TRel^+ \)

\( \text{by simp} \)

**next**

**case** (\textit{target T1 T2})

\( \begin{aligned} & \text{show} \ \forall SP \ SQ. \ SP \in S \ SourceTerm \ T1 \land SQ \in S \ SourceTerm \ T2 \rightarrow SP = SQ \\
& \land \forall SP \ TQ. \ SP \in S \ SourceTerm \ T1 \land TQ \in T \ SourceTerm \ T2 \rightarrow False \\
& \land \forall TP \ TQ. \ TP \in T \ SourceTerm \ T1 \land SQ \in S \ SourceTerm \ T2 \\
& \rightarrow (TP, \ [SQ]) \in (TRel \cup \{(T1, \ T2). \ \exists S. \ T1 = \llbracket S \rrbracket \land T2 = \llbracket S \rrbracket\})^+ \\
& \text{by simp-all} \end{aligned} \)

\( \text{assume} \ (T1, \ T2) \in TRel \)

\( \text{thus} \ \forall TP \ TQ. \ TP \in T \ SourceTerm \ T1 \land TQ \in T \ SourceTerm \ T2 \rightarrow (TP, \ TQ) \in TRel^+ \)

\( \text{by simp} \)

**next**

**case** (\textit{trans P Q R})

\( \begin{aligned} & \text{assume} \ A1: \ \forall SP \ SQ. \ SP \in S \ P \land SQ \in S \ Q \rightarrow SP = SQ \\
& \land A2: \ \forall SP \ TQ. \ SP \in S \ P \land TQ \in T \ Q \rightarrow False \\
& \land A3: \ \forall TP \ SQ. \ TP \in T \ P \land SQ \in S \ Q \\
& \rightarrow (TP, \ [SQ]) \in (TRel \cup \{(T1, \ T2). \ \exists S. \ T1 = \llbracket S \rrbracket \land T2 = \llbracket S \rrbracket\})^+ \\
& \land A4: \ \forall TP \ TQ. \ TP \in T \ P \land TQ \in T \ Q \rightarrow (TP, \ TQ) \in TRel^+ \\
& \land A5: \ \forall SQ \ SR. \ SQ \in S \ Q \land SR \in S \ R \rightarrow SQ = SR \\
& \land A6: \ \forall SQ \ TR. \ SQ \in S \ Q \land TR \in T \ R \rightarrow False \\
& \land A7: \ \forall TQ \ SR. \ TQ \in T \ Q \land SR \in S \ R \\
& \rightarrow (TQ, \ [SR]) \in (TRel \cup \{(T1, \ T2). \ \exists S. \ T1 = \llbracket S \rrbracket \land T2 = \llbracket S \rrbracket\})^+ \\
& \land A8: \ \forall TQ \ TR. \ TQ \in T \ Q \land TR \in T \ R \rightarrow (TQ, \ TR) \in TRel^+ \\
& \text{show} \ \forall SP \ SR. \ SP \in S \ P \land SR \in S \ R \rightarrow SP = SR \\
& \text{proof} \ (\textit{cases S}) \end{aligned} \)

\( \begin{aligned} & \text{case} \ (\textit{SourceTerm SQ}) \\
& \text{assume} \ SQ \in S \ Q \\
& \text{with} \ A1 \ A5 \ \text{show} \ \forall SP \ SR. \ SP \in S \ P \land SR \in S \ R \rightarrow SP = SR \\
& \text{by blast} \end{aligned} \)

**next**

**case** (\textit{target T1 Q})

\( \begin{aligned} & \text{assume} \ TQ \in T \ Q \\
& \text{with} \ A2 \ \text{show} \ ?thesis \\
& \text{by blast} \end{aligned} \)

**qed**

\( \text{show} \ \forall SP \ TR. \ SP \in S \ P \land TR \in T \ R \rightarrow False \\
\text{proof} \ (\textit{cases S}) \)

\( \begin{aligned} & \text{case} \ (\textit{SourceTerm SQ}) \\
& \text{assume} \ SQ \in S \ Q \\
& \text{with} \ A6 \ \text{show} \ ?thesis \\
& \text{by blast} \end{aligned} \)

**next**

**case** (\textit{target T1 Q})

90
assume $TQ \in T$ $Q$

with $A_2$ show $thesis$

by blast

qed

show $\forall TP SR. TP \in T \land SR \in S \land R$

$\rightarrow (TP, [SR]) \in (TRel \cup \{(T_1, T_2) \land \exists S. T_1 = [S] \land T_2 = [S]\})^+$

proof (cases $Q$)

next

proof (cases $Q$)

assume $TQ \in T$ $Q$

show $\forall TP SR. TP \in T \land SR \in S \land R$

$\rightarrow (TP, [SR]) \in (TRel \cup \{(T_1, T_2) \land \exists S. T_1 = [S] \land T_2 = [S]\})^+$

by blast

qed

lemma (in encoding) indRelTEQ-to-TRel:

fixes $TRel :: ('procT \times 'procT)$ set
and \( P Q \) :: ('procS, 'procT) Proc
assumes rel: \( P \leadsto T < \text{T} Rel > Q \)

shows \( \forall SP SQ. SP \in S P \land SQ \in S Q \)
\( \rightarrow ([SP], [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall SP TQ. SP \in S P \land TQ \in T Q \)
\( \rightarrow ([SP], TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP SQ. TP \in T P \land SQ \in S Q \)
\( \rightarrow (TP, [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP TQ. TP \in T P \land TQ \in T Q \)
\( \rightarrow (TP, TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

proof
–

have reflTRel: \( \forall S. ([S], [S]) \in \text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\} 
by auto

from refl show \( \forall SP SQ. SP \in S P \land SQ \in S Q \)
\( \rightarrow ([SP], [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall SP TQ. SP \in S P \land TQ \in T Q \)
\( \rightarrow ([SP], TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP SQ. TP \in T P \land SQ \in S Q \)
\( \rightarrow (TP, [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP TQ. TP \in T P \land TQ \in T Q \)
\( \rightarrow (TP, TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

proof
induct

–
case (encR S)

show \( \forall SP SQ. SP \in S \text{ SourceTerm} S \land SQ \in S \text{ TargetTerm} ([S]) \)
\( \rightarrow ([SP], [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP SQ. TP \in T \text{ SourceTerm} S \land SQ \in S \text{ TargetTerm} ([S]) \)
\( \rightarrow (TP, [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP TQ. TP \in T \text{ SourceTerm} S \land TQ \in T \text{ TargetTerm} ([S]) \)
\( \rightarrow (TP, TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by simp+

from reflTRel show \( \forall SP TQ. SP \in S \text{ SourceTerm} S \land TQ \in T \text{ TargetTerm} ([S]) \)
\( \rightarrow ([SP], TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by blast

next
–
case (encL S)

show \( \forall SP SQ. SP \in S \text{ TargetTerm} ([S]) \land SQ \in S \text{ SourceTerm} S \)
\( \rightarrow ([SP], [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall SP TQ. SP \in S \text{ TargetTerm} ([S]) \land TQ \in T \text{ SourceTerm} S \)
\( \rightarrow ([SP], TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP TQ. TP \in T \text{ TargetTerm} ([S]) \land TQ \in T \text{ SourceTerm} S \)
\( \rightarrow (TP, TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by simp+

from reflTRel show \( \forall TP SQ. TP \in T \text{ TargetTerm} ([S]) \land SQ \in S \text{ SourceTerm} S \)
\( \rightarrow (TP, [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by blast

next
–
case (target T1 T2)

show \( \forall SP SQ. SP \in S \text{ TargetTerm} T1 \land SQ \in S \text{ TargetTerm} T2 \)
\( \rightarrow ([SP], [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall SP TQ. SP \in S \text{ TargetTerm} T1 \land TQ \in T \text{ TargetTerm} T2 \)
\( \rightarrow ([SP], TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)
and \( \forall TP SQ. TP \in T \text{ TargetTerm} T1 \land SQ \in S \text{ TargetTerm} T2 \)
\( \rightarrow (TP, [SQ]) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by simp+

assume \( (T1, T2) \in \text{T} Rel \)

thus \( \forall TP TQ. TP \in T \text{ TargetTerm} T1 \land TQ \in T \text{ TargetTerm} T2 \)
\( \rightarrow (TP, TQ) \in (\text{T} Rel \cup \{(T1, T2). \exists S. T1 = [S] \land T2 = [S]\})^+ \)

by blast

next
–
case (trans P Q R)

assume \( A1: \forall SP SQ. SP \in S P \land SQ \in S Q \)

92
→ ([SP], [SQ]) ∈ (TRel ∪ {(T1, T2)}. ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A2: ∀S P. TQ. SP ∈ S P ∧ TQ ∈ T Q
→ ([SP], TQ) ∈ (TRel ∪ {(T1, T2)}. ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A3: ∀T P. SQ. TP ∈ T P ∧ SQ ∈ S Q
→ (TP, [SQ]) ∈ (TRel ∪ {(T1, T2)}. ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A4: ∀TP TQ. TP ∈ T P ∧ TQ ∈ T Q
→ (TP, TQ) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+
and A5: ∀SQ SR. SQ ∈ S Q ∧ SR ∈ S R
→ ([SQ], [SR]) ∈ (TRel ∪ {(T1, T2)}. ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A6: ∀SQ TR. SQ ∈ S Q ∧ TR ∈ T R
→ ([SQ], TR) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A7: ∀TQ SR. TQ ∈ T Q ∧ SR ∈ S R
→ (TQ, [SR]) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+ 
and A8: ∀TQ TR. TQ ∈ T Q ∧ TR ∈ T R
→ (TQ, TR) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+

show ∀SP TR. SP ∈ S P ∧ TR ∈ T R
→ ([SP], TR) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+

proof (cases Q)
case (SourceTerm SQ)
assume SQ ∈ S Q
with A1 A5 show ?thesis
by auto
next
case (TargetTerm TQ)
assume TQ ∈ T Q
with A2 A7 show ?thesis
by auto
qed
show ∀TP SR. TP ∈ T P ∧ SR ∈ S R
→ (TP, [SR]) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+

proof (cases Q)
case (SourceTerm SQ)
assume SQ ∈ S Q
with A3 A5 show ?thesis
by auto
next
case (TargetTerm TQ)
assume TQ ∈ T Q
with A4 A7 show ?thesis
by auto
qed
show ∀TP TR. TP ∈ T P ∧ TR ∈ T R
→ (TP, TR) ∈ (TRel ∪ {(T1, T2). ∃S. T1 = [S] ∧ T2 = [S])"}^+

proof (cases Q)
case (SourceTerm SQ)
assume SQ ∈ S Q
with A3 A6 show ?thesis
by auto
next
  case (TargetTerm TQ)
  assume TQ ∈ TQ
  with A4 A8 show ?thesis
    by auto
  qed
qed

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

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

lemma (in encoding) indRelRTPO-relates-source-target:
  fixes TRel :: ('procT × 'procT) set
  and S :: 'procS
  and T :: 'procT
  assumes pair: SourceTerm S ≲ [·]RT<TRel> TargetTerm T
  shows (TargetTerm ([S]), TargetTerm T) ∈ (indRelRTPO TRel)=
proof –
  from pair have ([S], T) ∈ TRel* using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond by simp
  hence [S] = T ∨ ([S], T) ∈ TRel* using rtrancl-eq-or-trancl[of [S] T TRel] by blast
  moreover have [S] = T ⇒ (TargetTerm ([S]), TargetTerm T) ∈ (indRelRTPO TRel)= by simp
  moreover have ([S], T) ∈ TRel* ⇒ (TargetTerm ([S]), TargetTerm T) ∈ (indRelRTPO TRel)= using transitive-closure-of-TRel-to-indRelRTPO[where TRel=TRel] by simp
  ultimately show (TargetTerm ([S]), TargetTerm T) ∈ (indRelRTPO TRel)= by blast
qed

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

lemma (in encoding-wrt-barbs) rel-with-target-impl-TRel-preserves-barbs:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: ('procS × 'procT) Proc × ('procS × 'procT) Proc) set

94
assumes preservation: rel-preservation-barbs Rel (STCalWB SWB TWB)
and targetInRel: \( \forall T1 T2. \ (T1, T2) \in TRel \rightarrow (\text{TargetTerm} \ T1, \text{TargetTerm} \ T2) \in \text{Rel} \)
shows rel-preservation-barbs TRel TWB

proof clarify

fix TP TQ a
assume \((TP, TQ) \in TRel\)
with targetInRel have \((\text{TargetTerm} \ TP, \text{TargetTerm} \ TQ) \in \text{Rel}\)
by blast

moreover assume \(TP \downarrow <TWB>a\)

hence \(\text{TargetTerm} \ TP \downarrow .a\)
by simp

ultimately have \(\text{TargetTerm} \ TQ \downarrow .a\)
using preservation preservation-of-barbs-in-barbed-encoding[where \(\text{Rel} = \text{Rel}\)]
by blast

thus \(TQ \downarrow <TWB>a\)
by simp

qed

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-preservation-barbs:
fixes TRel :: \((\text{procT} \times \text{procT})\) set
assumes preservation: rel-preservation-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
shows rel-preservation-barbs TRel TWB

using preservation
rel-with-target-impl-TRel-preservation-barbs[where \(\text{Rel} = \text{indRelRTPO} \ TRel \text{ and} \ TRel = TRel\)]
by (simp add: indRelRTPO.target)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-preservation-barbs:
fixes TRel :: \((\text{procT} \times \text{procT})\) set
assumes preservation: rel-preservation-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
shows rel-preservation-barbs TRel TWB

using preservation
rel-with-target-impl-TRel-preservation-barbs[where \(\text{Rel} = \text{indRelLTPO} \ TRel \text{ and} \ TRel = TRel\)]
by (simp add: indRelLTPO.target)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-preservation-barbs:
fixes TRel :: \((\text{procT} \times \text{procT})\) set
assumes preservation: rel-preservation-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
shows rel-preservation-barbs TRel TWB

using preservation
rel-with-target-impl-TRel-preservation-barbs[where \(\text{Rel} = \text{indRelTEQ} \ TRel \text{ and} \ TRel = TRel\)]
by (simp add: indRelTEQ.target)

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

proof clarify

fix TP TQ a TP'
assume \((TP, TQ) \in TRel\)
with targetInRel have \((\text{TargetTerm} \ TP, \text{TargetTerm} \ TQ) \in \text{Rel}\)
by blast

moreover assume \(TP \rightarrow (\text{Calculus} \ TWB)\)\* TP' \text{ and} \ TP' \downarrow <TWB>a

hence \(\text{TargetTerm} \ TP \downarrow .a\)
by blast

ultimately have \(\text{TargetTerm} \ TQ \downarrow .a\)
using preservation weak-preservation-of-barbs-in-barbed-encoding[where \(\text{Rel} = \text{Rel}\)]
by blast

thus \(TQ \downarrow <TWB>a\)
by simp
lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-weakly-preserves-bars:
  fixes TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-bars (indRelRTPO TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-bars TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-bars[where
        Rel=indRelRTPO TRel and TRel=TRel]
    by (simp add: indRelRTPO.target)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-weakly-preserves-bars:
  fixes TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-bars (indRelLTPO TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-bars TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-bars[where
        Rel=indRelLTPO TRel and TRel=TRel]
    by (simp add: indRelLTPO.target)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-weakly-preserves-bars:
  fixes TRel :: ('procT × 'procT) set
  assumes preservation: rel-weakly-preserves-bars (indRelTEQ TRel) (STCalWB SWB TWB)
  shows rel-weakly-preserves-bars TRel TWB
    using preservation rel-with-target-impl-TRel-weakly-preserves-bars[where
        Rel=indRelTEQ TRel and TRel=TRel]
    by (simp add: indRelTEQ.target)

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

lemma (in encoding-wrt-barbs) rel-with-target-impl-TRel-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)
  shows rel-reflects-barbs TRel TWB
  proof clarify
    fix TP TQ a
    assume (TP, TQ) ∈ TRel
    with targetInRel have (TargetTerm TP, TargetTerm TQ) ∈ Rel
      by blast
    moreover assume TQ↓<TWB>a
    hence TargetTerm TQ↓a
      by simp
    ultimately have TargetTerm TP↓a
      using reflection reflection-of-barbs-in-barbed-encoding[where Rel=Rel]
      by blast
    thus TP↓<TWB>a
      by simp
  qed

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-reflects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
  shows rel-reflects-barbs TRel TWB
    using reflection
    by (simp add: indRelRTPO.target)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-reflects-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
shows rel-reflects-barbs TRel TWB
  using reflection
  rel-with-target-impl-TRel-reflects-barbs[where Rel=indRelLTPO TRel and TRel=TRel]
  by (simp add: indRelLTPO.target)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-reflects-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
  shows rel-reflects-barbs TRel TWB
    using reflection
    rel-with-target-impl-TRel-reflects-barbs[where Rel=indRelTEQ TRel and TRel=TRel]
    by (simp add: indRelTEQ.target)

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

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-weakly-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-weakly-reflects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
  shows rel-weakly-reflects-barbs TRel TWB
  using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where
    Rel=indRelRTPO TRel and TRel=TRel]
  by (simp add: indRelRTPO.target)

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-weakly-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-weakly-reflects-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
  shows rel-weakly-reflects-barbs TRel TWB
  using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where
    Rel=indRelLTPO TRel and TRel=TRel]
  by (simp add: indRelLTPO.target)

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-weakly-reflects-barbs:
  fixes TRel :: ('procT × 'procT) set
  assumes reflection: rel-weakly-reflects-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
  shows rel-weakly-reflects-barbs TRel TWB
  using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where
    Rel=indRelTEQ TRel and TRel=TRel]
  by (simp add: indRelTEQ.target)

If indRelRTPO, indRelLTPO, or indRelTPO respects barbs then so does the corresponding target term relation.
lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
shows rel-respects-barbs TRel TWB
  using respection indRelRTPO-impl-TRel-preserves-barbs[where TRel=TRel]
  indRelRTPO-impl-TRel-reflects-barbs[where TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-respects-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
shows rel-respects-barbs TRel TWB
  using respection indRelLTPO-impl-TRel-preserves-barbs[where TRel=TRel]
  indRelLTPO-impl-TRel-reflects-barbs[where TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-respects-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
shows rel-respects-barbs TRel TWB
  using respection indRelTEQ-impl-TRel-preserves-barbs[where TRel=TRel]
  indRelTEQ-impl-TRel-reflects-barbs[where TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-weakly-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-weakly-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
shows rel-weakly-respects-barbs TRel TWB
  using respection indRelRTPO-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
  indRelRTPO-impl-TRel-weakly-reflects-barbs[where TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-weakly-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-weakly-respects-barbs (indRelLTPO TRel) (STCalWB SWB TWB)
shows rel-weakly-respects-barbs TRel TWB
  using respection indRelLTPO-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
  indRelLTPO-impl-TRel-weakly-reflects-barbs[where TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-weakly-respects-barbs:
fixes TRel :: ('procT × 'procT) set
assumes respection: rel-weakly-respects-barbs (indRelTEQ TRel) (STCalWB SWB TWB)
shows rel-weakly-respects-barbs TRel TWB
  using respection indRelTEQ-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
  indRelTEQ-impl-TRel-weakly-reflects-barbs[where TRel=TRel]
by blast

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

lemma (in encoding) rel-with-target-impl-transC-TRel-is-weak-reduction-simulation:
fixes TRel :: ('procT × 'procT) set
and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc set
assumes sim: weak-reduction-simulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  and trrel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel'
shows weak-reduction-simulation (TRel') Target
proof clarify
fix TP TQ TP'
assume (TP, TQ) ∈ TRel' and TP → Target* TP'
thus \( \exists TQ', TQ \rightsquigarrow \text{Target} \ast TQ' \land (TP', TQ') \in TRel^+ \)

proof (induct arbitrary: \( TP' \))

fix \( TQ \ TP' \)

assume \( (TP', TQ) \in TRel \)

with target have \((\text{TargetTerm} TP, \text{TargetTerm} TQ) \in \text{Rel})

by simp

moreover assume \( TP \mapsto \text{Target} \ast TP' \)

hence \( \text{TargetTerm} TP \mapsto \text{(STCal Source Target)} \ast (\text{TargetTerm} TP') \)

by \((\text{simp add: STCal-steps})\)

ultimately obtain \( Q' \) where \( A2: \text{TargetTerm} TQ \mapsto \text{(STCal Source Target)} \ast Q' \)

and \( A3: (\text{TargetTerm} TP', Q') \in \text{Rel} \)

using \( \text{sim} \)

by \( \text{blast} \)

from \( A2 \) obtain \( TQ' \) where \( A4: TQ \mapsto \text{Target} \ast TQ' \) and \( A5: TQ' \in TQ' \)

by \((\text{auto simp add: STCal-steps})\)

from \( A3 \) \( A5 \) trel have \((TP', TQ') \in TRel^+ \)

by \( \text{simp} \)

with \( A4 \) show \( \exists TQ'. TQ \mapsto \text{Target} \ast TQ' \land (TP', TQ') \in TRel^+ \)

by \( \text{blast} \)

next

case \((\text{step} TQ \ TR)\)

assume \( TP \mapsto \text{Target} \ast TP' \)

and \( \land TP', TP \mapsto \text{Target} \ast TP' \mapsto \exists TQ'. TQ \mapsto \text{Target} \ast TQ' \land (TP', TQ') \in TRel^+ \)

from this obtain \( TQ' \) where \( B1: TQ \mapsto \text{Target} \ast TQ' \) and \( B2: (TP', TQ') \in TRel^+ \)

by \( \text{blast} \)

assume \((TQ, TR) \in TRel \)

with target have \((\text{TargetTerm} TQ, \text{TargetTerm} TR) \in \text{Rel})

by \( \text{simp} \)

moreover from \( B1 \) have \( \text{TargetTerm} TQ \mapsto \text{(STCal Source Target)} \ast (\text{TargetTerm} TQ') \)

by \((\text{simp add: STCal-steps})\)

ultimately obtain \( R' \) where \( B3: \text{TargetTerm} TR \mapsto \text{(STCal Source Target)} \ast R' \) and \( B5: (\text{TargetTerm} TQ', R') \in \text{Rel} \)

using \( \text{sim} \)

by \( \text{blast} \)

from \( B3 \) obtain \( TR' \) where \( B5: TR' \in T R' \) and \( B6: TR \mapsto \text{Target} \ast TR' \)

by \((\text{auto simp add: STCal-steps})\)

from \( B4 \) \( B5 \) trel have \((TQ', TR') \in TRel^+ \)

by \( \text{simp} \)

with \( B2 \) have \((TP', TR') \in TRel^+ \)

by \( \text{simp} \)

with \( B6 \) show \( \exists TR'. TR \mapsto \text{Target} \ast TR' \land (TP', TR') \in TRel^+ \)

by \( \text{blast} \)

qed

qed

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

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

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

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

using \( \text{sim} \) \( \text{indRelRTPO} \)\text{.target[where TRel=TRel]} \text{indRelRTPO-to-TRel(4)[where TRel=TRel]} \rel-with-target-impl-transC-\text{TRel-is-weak-reduction-simulation[where Rel=indRelRTPO TRel and TRel=TRel]} \)

by \( \text{blast} \)

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

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

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

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

using \( \text{sim} \) \( \text{indRelLTPPO} \)\text{.target[where TRel=TRel]} \text{indRelLTPPO-to-TRel(4)[where TRel=TRel]} \rel-with-target-impl-transC-\text{TRel-is-weak-reduction-simulation[where Rel=indRelLTPPO TRel and TRel=TRel]} \)

99
lemma (in encoding) rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev:

fixes TRel :: ('procS × 'procT) set
assumes sim: weak-reduction-simulation (Rel⁻¹) (STCal Source Target)
and target: ∀T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
and trel: ∀T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel⁺
shows weak-reduction-simulation ((TRel⁺)⁻¹) Target

proof clarify

fix TP TQ TP′
assume (TQ, TP) ∈ TRel⁺
moreover assume TP → Target* TP′
ultimately show ∃ TQ′. TQ → Target* TQ′ ∧ (TP′, TQ′) ∈ (TRel⁺)⁻¹

proof (induct arbitrary: TP′)
fix TP TP′
assume (TQ, TP) ∈ TRel
with target have (TargetTerm TP, TargetTerm TQ) ∈ Rel⁻¹
  by simp
moreover assume TP → Target* TP′
hence TargetTerm TP → (STCal Source Target)* (TargetTerm TP′)
  by (simp add: STCal-steps)
ultimately obtain Q′ where A2: TargetTerm TQ → (STCal Source Target)* Q′
  and A3: (TargetTerm TP′, Q′) ∈ Rel⁻¹
  using simp
  by blast
from A2 obtain TQ′ where A4: TQ → Target* TQ′ and A5: TQ′ ∈ T Q′
  by (auto simp add: STCal-steps(2))
from A3 A5 trel have (TP′, TQ′) ∈ (TRel⁺)⁻¹
  by simp
with A4 show ∃ TQ′. TQ → Target* TQ′ ∧ (TP′, TQ′) ∈ (TRel⁺)⁻¹
  by blast

next

case (step TR TP TP′)
assume TP → Target* TP′
hence TargetTerm TP → (STCal Source Target)* (TargetTerm TP′)
  by (simp add: STCal-steps)
moreover assume (TR, TP) ∈ TRel
with target have (TargetTerm TP, TargetTerm TR) ∈ Rel⁻¹
  by simp
ultimately obtain R′ where B1: TargetTerm TR → (STCal Source Target)* R′
  and B2: (TargetTerm TP′, R′) ∈ Rel⁻¹
  using simp
  by blast
from B1 obtain TR′ where B3: TR′ ∈ T R′ and B4: TR → Target* TR′
  by (auto simp add: STCal-steps)
assume TR′ ∧ TR. TR → Target* TR′ → ∃ TQ′. TQ → Target* TQ′ ∧ (TR′, TQ′) ∈ (TRel⁺)⁻¹
with B4 obtain TQ′ where B5: T Q → Target* TQ′ and B6: (TR′, TQ′) ∈ (TRel⁺)⁻¹
  by blast
from B6 have (TQ′, TR′) ∈ TRel⁺
  by simp
moreover from B2 B3 trel have (TR′, TP′) ∈ TRel⁺
  by simp
ultimately have (TP′, TQ′) ∈ (TRel⁺)⁻¹
  by simp
with B5 show ∃ TQ′. TQ → Target* TQ′ ∧ (TP′, TQ′) ∈ (TRel⁺)⁻¹
  by blast
qed
qed

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev:
fixes $T_{Rel} :: ('procT \times 'procT)$ set
assumes sim: weak-reduction-simulation $((\text{indRelRTPO } T_{Rel})^{-1})$ (STCal Source Target)
shows weak-reduction-simulation $((T_{Rel}^+)^{-1})$ Target
using sim $\text{indRelRTPO}.\text{target}[\text{where } T_{Rel}=\text{TRel}]$ $\text{indRelRTPO-to-TRel(4)}[\text{where } T_{Rel}=\text{TRel}]$
$\text{rel-with-target-impl-transC}.\text{TRel-is-weak-reduction-simulation-rev}[\text{where } \text{Rel}=\text{indRelRTPO } T_{Rel}$ and $T_{Rel}=\text{TRel}]$
by blast

lemma (in encoding) $\text{indRelLTPO-impl-TRel-is-weak-reduction-simulation-rev}$:
fixes $T_{Rel} :: ('procT \times 'procT)$ set
assumes sim: weak-reduction-simulation $((\text{indRelLTPO } T_{Rel})^{-1})$ (STCal Source Target)
shows weak-reduction-simulation $((T_{Rel}^+)^{-1})$ Target
using sim $\text{indRelLTPO}.\text{target}[\text{where } T_{Rel}=\text{TRel}]$ $\text{indRelLTPO-to-TRel(4)}[\text{where } T_{Rel}=\text{TRel}]$
$\text{rel-with-target-impl-reflC}.\text{transC}.\text{TRel-is-weak-reduction-simulation}$
by blast

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

proof clarify
fix $T T \ T Q T P'$
assume $(T, T Q) \in T_{Rel}^*$ and $T P \longrightarrow \text{Target} \ast T P'$
thus $\exists T Q'. T Q' \longrightarrow \text{Target} \ast T Q' \land (T P', T Q') \in T_{Rel}^*$
proof (induct arbitrary: T P')
fix T P'
assume $T P \longrightarrow \text{Target} \ast T P'$
moreover have $(T P', T P') \in T_{Rel}^*$
by simp
ultimately show $\exists T Q'. T P \longrightarrow \text{Target} \ast T Q' \land (T P', T Q') \in T_{Rel}^*$
by blast
next
case (step T Q T R)
assume $T P \longrightarrow \text{Target} \ast T P'$
and $\land T P', T P \longrightarrow \text{Target} \ast T P' \Longrightarrow \exists T Q'. T Q \longrightarrow \text{Target} \ast T Q' \land (T P', T Q') \in T_{Rel}^*$
from this obtain $T Q'$ where $B1$: $T Q \longrightarrow \text{Target} \ast T Q' \land (T P', T Q') \in T_{Rel}^*$
by blast
assume $(T Q, T R) \in T_{Rel}$
with target have (TargetTerm T Q, TargetTerm T R) \in \text{Rel}
by simp
moreover from B1 have TargetTerm T Q \longrightarrow (STCal Source Target)^* (TargetTerm T Q')
by (simp add: STCal-steps)
ultimately obtain $R'$ where $B3$: TargetTerm T R \longrightarrow (STCal Source Target)^* R'
and $B4$: (TargetTerm T Q', R') \in \text{Rel}$
using sim
by blast
from B3 obtain TR' where $B5$: TR' \in T R' and $B6$: TR \longrightarrow Target \ast TR'
by (auto simp add: STCal-steps)
from B4 B5 trel have $(T Q', R') \in T_{Rel}^*$
by simp
with $B2$ have $(T P', TR') \in T_{Rel}^*$
by simp
with $B6$ show $\exists TR'. T R \longrightarrow \text{Target} \ast TR' \land (T P', TR') \in T_{Rel}^*$
by blast
qed
qed
lemma (in encoding) indRelTEQ-impl-TRel-is-weak-reduction-simulation:
fixes TRel :: ('procT × 'procT) set
assumes sim: weak-reduction-simulation (indRelTEQ TRel) (STCal Source Target)
shows weak-reduction-simulation (TRel+) Target
  using sim indRelTEQ.target[where TRel=TRel] indRelTEQ-to-TRel(4)[where TRel=TRel]
  trans-closure-of-TRel-refl-cond
rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation[where Rel=indRelTEQ TRel and TRel=TRel]
by blast

lemma (in encoding) rel-with-target-impl-transC-TRel-is-strong-reduction-simulation:
fixes TRel :: ('procT × 'procT) set
and Rel :: ('(procS, 'procT) Proc × ('procS, 'procT) Proc) set
assumes sim: strong-reduction-simulation Rel (STCal Source Target)
  and target: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  and trel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel+
shows strong-reduction-simulation (TRel+) Target
proof clarify
  fix TP TQ TP'
  assume (TP, TQ) ∈ TRel and TP → Target TP'
  thus ∃ TQ'. TQ → Target TQ' ∧ (TP', TQ') ∈ TRel+
proof (induct arbitrary: TP')
  fix TQ TP'
  assume (TP, TQ) ∈ TRel
  with target have (TargetTerm TP, TargetTerm TQ) ∈ Rel
  by simp
  moreover assume TP → Target TP'
  hence TargetTerm TP →(STCal Source Target) (TargetTerm TP')
  by (simp add: STCal-step)
ultimately obtain Q' where A2: TargetTerm TQ →(STCal Source Target) Q'
  and A3: (TargetTerm TP', Q') ∈ Rel
    using sim
    by blast
  from A2 obtain TQ' where A4: TQ → Target TQ' and A5: TQ' ∈ T Q'
    by (auto simp add: STCal-step)
  from A3 A5 trel have (TP', TQ') ∈ TRel+
    by simp
  with A4 show ∃ TQ'. TQ → Target TQ' ∧ (TP', TQ') ∈ TRel+
    by blast
next
  case (step TQ TR)
  assume TP → Target TP'
  and ∃ TP', TP → Target TP' → ∃ TQ', TQ → Target TQ' ∧ (TP', TQ') ∈ TRel+
  from this obtain TQ' where B1: TQ → Target TQ' and B2: (TP', TQ') ∈ TRel+
  by blast
  assume (TQ, TR) ∈ TRel
  with target have (TargetTerm TQ, TargetTerm TR) ∈ Rel
  by simp
  moreover from B1 have TargetTerm TQ →(STCal Source Target) (TargetTerm TQ')
  by (simp add: STCal-step)
ultimately obtain R' where B3: TargetTerm TR →(STCal Source Target) R'
  and B4: (TargetTerm TQ', R') ∈ Rel
    using sim
    by blast
  from B3 obtain TR' where B5: TR' ∈ T R' and B6: TR → Target TR'
    by (auto simp add: STCal-step)
  from B4 B5 trel have (TQ', TR') ∈ TRel+
    by simp
  with B2 have (TP', TR') ∈ TRel+
    by simp
  with B6 show ∃ TR'. TR → Target TR' ∧ (TP', TR') ∈ TRel+
by blast
qed

lemma (in encoding) indRelRTPO-impl-TRel-is-strong-reduction-simulation:
fixes TRel :: ('procT × 'procT) set
assumes sim: strong-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
shows strong-reduction-simulation (TRel+ ) Target
  using sim indRelRTPO.target[where TRel= TRel] indRelRTPO-to-TRel(4)[where TRel= TRel]
rel-with-target-impl-transC-TRel-is-strong-reduction-simulation[where
  Rel= indRelRTPO TRel and TRel= TRel]
by blast

lemma (in encoding) indRelLTPO-impl-TRel-is-strong-reduction-simulation:
fixes TRel :: ('procT × 'procT) set
assumes sim: strong-reduction-simulation (indRelLTPO TRel) (STCal Source Target)
shows strong-reduction-simulation (TRel+) Target
  using sim indRelLTPO.target[where TRel= TRel] indRelLTPO-to-TRel(4)[where TRel= TRel]
rel-with-target-impl-transC-TRel-is-strong-reduction-simulation[where
  Rel= indRelLTPO TRel and TRel= TRel]
by blast

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

using sim

by blast

from B1 obtain TP' where B3: TP' \in T P and B4: TP \rightarrow\text{Target TP'}

by (auto simp add: \text{STCal-step})

assume \(\land TP', TP \rightarrow\text{Target TP'} \implies \exists TQ'. TQ \rightarrow\text{Target TQ'} \land (TP', TQ') \in (TRel^+)\)

with B4 obtain TQ' where B5: TQ \rightarrow\text{Target TQ'} and B6: (TP', TQ') \in (TRel^+)\)

by blast

from B2 B3 trel have (TP', TR') \in TRel^+

by simp

with B6 have (TR', TQ') \in (TRel^+)\)

by simp

with B5 show \(\exists TQ'. TQ \rightarrow\text{Target TQ'} \land (TR', TQ') \in (TRel^+)\)

by blast

qed

qed

lemma (in encoding) \text{indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev}:

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

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

shows strong-reduction-simulation ((\text{indRelRTPO TRel})\) Target

using sim \text{indRelRTPO-targ}[\text{where TRel=TRel}] \text{indRelRTPO-to-Trel(4)[where TRel=TRel]}\)

rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev[\text{where Rel=indRelRTPO TRel and TRel=TRel}]\)

by blast

lemma (in encoding) \text{indRelLTOPO-impl-TRel-is-strong-reduction-simulation-rev}:

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

assumes sim: strong-reduction-simulation ((\text{indRelLTOPO TRel})\)

shows strong-reduction-simulation ((\text{indRelLTOPO TRel})\) Target

using sim \text{indRelLTOPO-targ}[\text{where TRel=TRel}] \text{indRelLTOPO-to-Trel(4)[where TRel=TRel]}\)

rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev[\text{where Rel=indRelLTOPO TRel and TRel=TRel}]\)

by blast

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

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

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

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

\(\rightarrow (T1, T2) \in \text{TRel}^\ast\)

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

proof clarify

fix TP TQ TP'

assume \((TP, TQ) \in \text{TRel}^\ast and TP \rightarrow\text{Target TP'}\)

thus \(\exists TQ'. TQ \rightarrow\text{Target TQ'} \land (TP', TQ') \in \text{TRel}^\ast\)

proof (induct arbitrary: TP')

fix TP'

assume TP \rightarrow\text{Target TP'}

moreover have \((TP', TP') \in \text{TRel}^\ast\)

by simp

ultimately show \(\exists TQ'. TP \rightarrow\text{Target TQ'} \land (TP', TQ') \in \text{TRel}^\ast\)

by blast

next

case \(\text{step} TP \ TR TP'\)

assume TP \rightarrow\text{Target TP'}

and \(\land TP', TP \rightarrow\text{Target TP'} \implies \exists TQ'. TQ \rightarrow\text{Target TQ'} \land (TP', TQ') \in \text{TRel}^\ast\)

from this obtain TQ' where B1: TQ \rightarrow\text{Target TQ'} and B2: (TP', TQ') \in \text{TRel}^\ast

by blast

assume (TQ, TR) \in \text{TRel}
with target have \((\text{TargetTerm } TQ, \text{TargetTerm } TR) \in \text{Rel}\)
by simp
moreover from \(B1\) have \(\text{TargetTerm } TQ \rightarrow (\text{STCal Source Target}) (\text{TargetTerm } TQ')\)
by (simp add: STCal-step)
ultimately obtain \(R'\) where \(B3: \text{TargetTerm } TR \rightarrow (\text{STCal Source Target}) R'\)
and \(B4: (\text{TargetTerm } TQ', R') \in \text{Rel}\)
using sim
by blast
from \(B3\) obtain \(TR'\) where \(B5: TR' \in T R'\) and \(B6: TR \rightarrow Target \ TR'\)
by (auto simp add: STCal-step)
from \(B4\) \(B5\) \texttt{tre} have \((TQ', TR') \in T\text{Rel}^*\)
by simp
with \(B2\) have \((TP', TR') \in TR^*\)
by simp
with \(B6\) show \(\exists \ TR'. \ TR \rightarrow Target \ TR' \land (TP', TR') \in TR^*\)
by blast
qed

\textbf{lemma (in encoding)} \(\text{indRelTEQ-impl-TRel-is-strong-reduction-simulation}\):
fixes \(\text{TRel} \:: \left( \text{procT} \times \text{procT} \right) \text{set}\)
assumes \(\text{sim} \:: \text{strong-reduction-simulation} \left( \text{indRelTEQ } \text{TRel} \right) (\text{STCal Source Target})\)
shows \(\text{strong-reduction-simulation} \left( \text{TRel}^* \right) \text{Target}\)
using \(\text{sim indRelTEQ.target[where } \text{TRel=}TRel[\text{indRelTEQ-to-TRel(4)}][where } \text{TRel=}TRel[\text{trans-closure-of-TRel-refl-cond}\text{rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation[where } \text{Rel=}\text{indRelTEQ } \text{TRel and } \text{TRel=}\text{TRel}]\)
by blast

\textbf{lemma (in encoding-wrt-barbs)} \(\text{indRelRTPO-impl-TRel-is-weak-barbed-simulation}\):
fixes \(\text{TRel} \:: \left( \text{procT} \times \text{procT} \right) \text{set}\)
assumes \(\text{sim} \:: \text{weak-barbed-simulation} \left( \text{indRelRTPO } \text{TRel} \right) (\text{STCalWB SWB TWB})\)
shows \(\text{weak-barbed-simulation} \left( \text{TRel}^* \right) \text{TWB}\)
proof
from \(\text{sim}\) show \(\text{weak-reduction-simulation} \left( \text{TRel}^* \right) \text{(Calculus TWB)}\)
using \(\text{indRelRTPO-impl-TRel-is-weak-reduction-simulation[where } \text{TRel=}TRel[\text{weak-preservation-of-barbs-and-closures(2)[where } \text{Rel=}\text{TRel and } \text{CWB=}\text{TWB}]\}
by blast
next
from \(\text{sim}\) show \(\text{rel-weakly-preserves-barbs} \left( \text{TRel}^* \right) \text{TWB}\)
using \(\text{indRelRTPO-impl-TRel-weakly-preserves-barbs[where } \text{TRel=}TRel[\text{weak-preservation-of-barbs-and-closures(2)[where } \text{Rel=}\text{TRel and } \text{CWB=}\text{TWB}]\}
by blast
qed

\textbf{lemma (in encoding-wrt-barbs)} \(\text{indRelLTPO-impl-TRel-is-weak-barbed-simulation}\):
fixes \(\text{TRel} \:: \left( \text{procT} \times \text{procT} \right) \text{set}\)

assumes sim: weak-barbed-simulation (indRelTEQ TRel) (STCalWB SWB TWB)
shows weak-barbed-simulation (TRel\star) TWB
proof
  from sim show weak-reduction-simulation (TRel\star) (Calculus TWB)
    using indRelTEQ-impl-TRel-is-weak-reduction-simulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from sim show rel-weakly-preserves-barbs (TRel\star) TWB
    using indRelTEQ-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
    weak-preservation-of-barbs-and-closures(3)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-strong-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-barbed-simulation (indRelRTPO TRel) (STCalWB SWB TWB)
  shows strong-barbed-simulation (TRel\star) TWB
proof
  from sim refl show strong-reduction-simulation (TRel\star) (Calculus TWB)
    using indRelRTPO-impl-TRel-is-strong-reduction-simulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from sim show rel-preserves-barbs (TRel\star) TWB
    using indRelRTPO-impl-TRel-preserves-barbs[where TRel=TRel]
    preservation-of-barbs-and-closures(2)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-strong-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-barbed-simulation (indRelLTPO TRel) (STCalWB SWB TWB)
  shows strong-barbed-simulation (TRel\star) TWB
proof
  from sim refl show strong-reduction-simulation (TRel\star) (Calculus TWB)
    using indRelLTPO-impl-TRel-is-strong-reduction-simulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from sim show rel-preserves-barbs (TRel\star) TWB
    using indRelLTPO-impl-TRel-preserves-barbs[where TRel=TRel]
    preservation-of-barbs-and-closures(2)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-strong-barbed-simulation:
  fixes TRel :: ('procT × 'procT) set
  assumes sim: strong-barbed-simulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows strong-barbed-simulation (TRel\star) TWB
proof
  from sim refl show strong-reduction-simulation (TRel\star) (Calculus TWB)
    using indRelTEQ-impl-TRel-is-strong-reduction-simulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from sim show rel-preserves-barbs (TRel\star) TWB
    using indRelTEQ-impl-TRel-preserves-barbs[where TRel=TRel]
    preservation-of-barbs-and-closures(3)[where Rel=TRel and CWB=TWB]
    by blast
qed

If indRelRTPO, indRelLTPO, or indRelTEQ is a contrasimulation then so is the corresponding target term relation.
lemma (in encoding) rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation:
 fixes TRel :: (‘procT × ‘procT) set
 assumes conSim: weak-reduction-contrasimulation Rel (STCal Source Target)
 and target: ∀ T1 T2. (T1, T2) ∈ TRel →→ (TargetTerm T1, TargetTerm T2) ∈ Rel
 and trel: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel →→ (T1, T2) ∈ TRel+
 shows weak-reduction-contrasimulation (TRel+) Target
proof clarify
 fix TP TQ TP'
 assume (TP, TQ) ∈ TRel+ and TP →→ Target* TP'
 thus ∃ TQ'. TQ →→ Target* TQ' ∧ (TQ', TP') ∈ TRel+
proof (induct arbitrary: TP')
 fix TQ TP'
 assume (TP, TQ) ∈ TRel
 with target have (TargetTerm TP, TargetTerm TQ) ∈ Rel
 by simp
 moreover assume TP →→ Target* TP'
 hence TargetTerm TP →→ (STCal Source Target)* (TargetTerm TP')
 by (simp add: STCal-steps)
 ultimately obtain Q' where A2: TargetTerm TQ →→ (STCal Source Target)* Q'
 and A3: (Q', TargetTerm TP') ∈ Rel
 using conSim
 by blast
 from A2 obtain TQ' where A4: TQ →→ Target* TQ' and A5: TQ' ∈ T Q'
 by (auto simp add: STCal-steps)
 from A3 A5 trel have (TQ', TP') ∈ TRel+
 by simp
 with A4 show ∃ TQ'. TQ →→ Target* TQ' ∧ (TQ', TP') ∈ TRel+
 by blast
 next
 case (step TQ TR)
 assume TP →→ Target* TP'
 and ∧ TP'. TP →→ Target* TP' →→ ∃ TQ'. TQ →→ Target* TQ' ∧ (TQ', TP') ∈ TRel+
 from this obtain TQ' where B1: TQ →→ Target* TQ' and B2: (TQ', TP') ∈ TRel+
 by blast
 assume (TQ, TR) ∈ TRel
 with target have (TargetTerm TP, TargetTerm TR) ∈ Rel
 by simp
 moreover from B1 have TargetTerm TQ →→ (STCal Source Target)* (TargetTerm TQ')
 by (simp add: STCal-steps)
 ultimately obtain R' where B3: TargetTerm TR →→ (STCal Source Target)* R'
 and B4: (R', TargetTerm TQ') ∈ Rel
 using conSim
 by blast
 from B3 obtain TR' where B5: TR' ∈ T R' and B6: TR →→ Target* TR'
 by (auto simp add: STCal-steps)
 from B4 B5 trel have (TR', TQ') ∈ TRel+
 by simp
 from this B2 have (TR', TP') ∈ TRel+
 by simp
 with B6 show ∃ TR'. TR →→ Target* TR' ∧ (TR', TP') ∈ TRel+
 by blast
 qed
 qed

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation:
 fixes TRel :: (‘procT × ‘procT) set
 assumes conSim: weak-reduction-contrasimulation (indRelRTPO TRel) (STCal Source Target)
 shows weak-reduction-contrasimulation (TRel+) Target
 using conSim indRelRTPO.target[where TRel= TRel] indRelRTPO-to-TRel(4)[where TRel= TRel]
 rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation[where
\[ \text{Rel} = \text{indRelRTPO TRel and TRel=TRel} \]

by blast

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

fixes TRel :: ('procT × 'procT) set
assumes conSim: weak-reduction-contrasimulation (indRelLTPO TRel) (\text{STCal Source Target})
shows weak-reduction-contrasimulation (TRel) Target

using conSim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(4)[where TRel=TRel]
rel-with-target-impl-transC-TRel-is-weak-reduction-contrasimulation[where
Rel=indRelLTPO TRrel and TRel=TRel]

by blast

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

fixes TRel :: ('procT × 'procT) set
and Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
assumes conSim: weak-reduction-contrasimulation Rel (\text{STCal Source Target})
and target: \( \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm T1, TargetTerm T2}) \in Rel \)
and trel: \( \forall T1 T2. (\text{TargetTerm T1, TargetTerm T2}) \in Rel \rightarrow (T1, T2) \in TRel^* \)
shows weak-reduction-contrasimulation (TRel) Target

proof clarify

fix TP TQ TP'
assume (TP, TQ) \in TRel and TP \rightarrow TARGET* TP'
thus \( \exists TQ'. TQ \rightarrow TARGET* TQ' \land (TQ', TP') \in TRel^* \)

proof (induct arbitrary: TP')

fix TP'
assume TP \rightarrow TARGET* TP'
moreover have (TP', TP') \in TRel^*

by simp
ultimately show \( \exists TQ'. TP \rightarrow TARGET* TQ' \land (TQ', TP') \in TRel^* \)

by blast

next

case (step TQ TR)
assume TP \rightarrow TARGET* TP'

and \( \land TP'. TP \rightarrow TARGET* TP' \rightarrow \exists TQ'. TQ \rightarrow TARGET* TQ' \land (TQ', TP') \in TRel^* \)

from this obtain TQ' where B1: TQ \rightarrow TARGET* TQ' and B2: (TQ', TP') \in TRel^*

by blast

assume (TQ, TR) \in TRel
with target have (TargetTerm TQ, TargetTerm TR) \in Rel

by simp

moreover from B1 have TargetTerm TQ \rightarrow (\text{STCal Source Target})* (TargetTerm TQ')

by (simp add: STCal-steps)

ultimately obtain R' where B3: TargetTerm TR \rightarrow (\text{STCal Source Target})* R'

and B4: (R', TargetTerm TQ') \in Rel

using conSim

by blast

from B3 obtain TR' where B5: TR' \in T R' and B6: TR \rightarrow TARGET* TR'

by (auto simp add: STCal-steps)

from B4 B5 trel have (TR', TQ') \in TRel^*

by simp

from this B2 have (TR', TP') \in TRel^*

by simp

with B6 show \( \exists TR'. TR \rightarrow TARGET* TR' \land (TR', TP') \in TRel^* \)

by blast

qed

d qed

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

fixes TRel :: ('procT × 'procT) set
assumes conSim: weak-reduction-contrasimulation (indRelTEQ TRel) (\text{STCal Source Target})
shows weak-reduction-contrasimulation (TRel) Target

using conSim indRelTEQ[target[where TRel=TRel] indRelTEQ-to-TRel(4)[where TRel=TRel]
trans-closure-of-TRel-refl-cond
rel-with-target-impl-reflC-trnsC-TRel-is-weak-reduction-contrasimulation[where
Rel=indRelTEQ TRel and TRel=TRel]
by blast

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation:
fixes TRel :: (′procT × ′procT) set
assumes conSim: weak-barbed-contrasimulation (indRelRTPO TRel) (STCalWB SWB TWB)
shows weak-barred-contrasimulation (TRel+) TWB
proof
from conSim show weak-reduction-contrasimulation (TRel+) (Calculus TWB)
using indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation[where TRel=TRel]
by (simp add: STCalWB-def calS calT)
next
from conSim show rel-weakly-preserves-barbs (TRel+) TWB
using indRelRTPO-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
weak-preservation-of-barbs-and-closures(2)[where Rel=TRel and CWB=TWB]
by blast
qed

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation:
fixes TRel :: (′procT × ′procT) set
assumes conSim: weak-barbed-contrasimulation (indRelLTPO TRel) (STCalWB SWB TWB)
shows weak-barred-contrasimulation (TRel+) TWB
proof
from conSim show weak-reduction-contrasimulation (TRel+) (Calculus TWB)
using indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation[where TRel=TRel]
by (simp add: STCalWB-def calS calT)
next
from conSim show rel-weakly-preserves-barbs (TRel+) TWB
using indRelLTPO-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
weak-preservation-of-barbs-and-closures(2)[where Rel=TRel and CWB=TWB]
by blast
qed

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation:
fixes TRel :: (′procT × ′procT) set
assumes conSim: weak-barbed-contrasimulation (indRelTEQ TRel) (STCalWB SWB TWB)
shows weak-barred-contrasimulation (TRel+) TWB
proof
from conSim show weak-reduction-contrasimulation (TRel+) (Calculus TWB)
using indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation[where TRel=TRel]
by (simp add: STCalWB-def calS calT)
next
from conSim show rel-weakly-preserves-barbs (TRel+) TWB
using indRelTEQ-impl-TRel-weakly-preserves-barbs[where TRel=TRel]
weak-preservation-of-barbs-and-closures(3)[where Rel=TRel and CWB=TWB]
by blast
qed

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

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-coupled-simulation:
fixes TRel :: (′procT × ′procT) set
assumes couSim: weak-reduction-coupled-simulation (indRelRTPO TRel) (STCal Source Target)
shows weak-reduction-coupled-simulation (TRel+) Target
proof
from couSim show weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation
refl indRelRTPO-impl-TRel-is-weak-reduction-simulation[where TRel=TRel]
indRelRTPO-impl-TRel-is-weak-reduction-contrasimulation[where TRel=TRel]
by blast

qed
lemma (in encoding) \(\text{indRelLTPO-impl-TRel-is-weak-reduction-coupled-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{assumes } \text{couSim: weak-reduction-coupled-simulation (indRelLTPO TRel) (STCal Source Target)}\]
\[\text{shows } \text{weak-reduction-coupled-simulation (TRel\textsuperscript{+}) Target}\]
\[\text{using } \text{couSim weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation}\]
\[\text{refl indRelLTPO-impl-TRel-is-weak-reduction-simulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{indRelLTPO-impl-TRel-is-weak-reduction-contrasimulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{by blast}\]

lemma (in encoding) \(\text{indRelTEQ-impl-TRel-is-weak-reduction-coupled-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{assumes } \text{couSim: weak-reduction-coupled-simulation (indRelTEQ TRel) (STCal Source Target)}\]
\[\text{shows } \text{weak-reduction-coupled-simulation (TRel\textsuperscript{+}) Target}\]
\[\text{using } \text{couSim weak-reduction-coupled-simulation-versus-simulation-and-contrasimulation}\]
\[\text{refl indRelTEQ-impl-TRel-is-weak-reduction-simulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{indRelTEQ-impl-TRel-is-weak-reduction-contrasimulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{by blast}\]

lemma (in encoding-wrt-barbs) \(\text{indRelRTPO-impl-TRel-is-weak-barbed-coupled-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{assumes } \text{couSim: weak-barbed-coupled-simulation (indRelRTPO TRel) (STCalWB SWB TWB)}\]
\[\text{shows } \text{weak-barbed-coupled-simulation (TRel\textsuperscript{+}) TWB}\]
\[\text{using } \text{couSim weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation}\]
\[\text{refl indRelRTPO-impl-TRel-is-weak-barbed-simulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{indRelRTPO-impl-TRel-is-weak-barbed-contrasimulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{by blast}\]

lemma (in encoding-wrt-barbs) \(\text{indRelLTPO-impl-TRel-is-weak-barbed-coupled-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{assumes } \text{couSim: weak-barbed-coupled-simulation (indRelLTPO TRel) (STCalWB SWB TWB)}\]
\[\text{shows } \text{weak-barbed-coupled-simulation (TRel\textsuperscript{+}) TWB}\]
\[\text{using } \text{couSim weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation}\]
\[\text{refl indRelLTPO-impl-TRel-is-weak-barbed-simulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{indRelLTPO-impl-TRel-is-weak-barbed-contrasimulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{by blast}\]

lemma (in encoding-wrt-barbs) \(\text{indRelTEQ-impl-TRel-is-weak-barbed-coupled-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{assumes } \text{couSim: weak-barbed-coupled-simulation (indRelTEQ TRel) (STCalWB SWB TWB)}\]
\[\text{shows } \text{weak-barbed-coupled-simulation (TRel\textsuperscript{+}) TWB}\]
\[\text{using } \text{couSim weak-barbed-coupled-simulation-versus-simulation-and-contrasimulation}\]
\[\text{refl indRelTEQ-impl-TRel-is-weak-barbed-simulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{indRelTEQ-impl-TRel-is-weak-barbed-contrasimulation[where } T\text{Rel}=T\text{Rel}]\]
\[\text{by blast}\]

If \(\text{indRelRTPO}, \text{indRelLTPO},\) or \(\text{indRelTEQ}\) is a correspondence simulation then so is the corresponding target term relation.

lemma (in encoding) \(\text{rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation:}\)
\[\text{fixes } T\text{Rel} :: (\text{\texttt{proc}T} \times \text{\texttt{proc}T})\text{ set}\]
\[\text{and } \text{Rel} :: (\text{\texttt{proc}S}, \text{\texttt{proc}T})\text{ Proc } \times (\text{\texttt{proc}S}, \text{\texttt{proc}T})\text{ Proc}\text{ set}\]
\[\text{assumes } \text{couSim: weak-reduction-correspondence-simulation Rel (STCal Source Target)}\]
\[\text{and } \text{target: } \forall T1 T2. (T1, T2) \in T\text{Rel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\]
\[\text{and } \text{trel: } \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in T\text{Rel}\textsuperscript{+}\]
\[\text{shows } \text{weak-reduction-correspondence-simulation (TRel\textsuperscript{+}) Target}\]
\[\text{proof}\]
\[\text{from } \text{corSim target trel have } A: \text{weak-reduction-simulation (TRel\textsuperscript{+}) Target}\]
\[\text{using } \text{rel-with-target-impl-transC-TRel-is-weak-reduction-simulation[where } T\text{Rel}=T\text{Rel}\]
\[\text{and } \text{Rel}=\text{Rel}\]
\[\text{by blast}\]
moreover have \( \forall P \, Q \, Q' \). \((P, Q) \in T\text{Rel}^+ \land Q \longrightarrow T\text{Target} \land Q'\)

\[ \rightarrow (\exists P'' \, Q''. \ P \longrightarrow T\text{Target} \land P'' \land Q' \longrightarrow T\text{Target} \land Q'' \land (P'', Q'') \in T\text{Rel}^+) \]

**proof** clarify

fix \( T \), \( Q \), \( T' \)

assume \((T, T') \in T\text{Rel}^+ \) and \( T \longrightarrow T\text{Target} \land T' \)

thus \( \exists T' \, T'' \). \( T' \longrightarrow T\text{Target} \land T'' \land T' \longrightarrow T\text{Target} \land T'' \land (T'', T'') \in T\text{Rel}^+ \)

**proof** (induct arbitrary: \( T' \))

fix \( T \), \( Q \), \( T' \)

assume \((T, T') \in T\text{Rel} \)

with \( \text{Target} \) have \((\text{TargetTerm } T, \text{TargetTerm } T') \in \text{Rel} \)

by blast

moreover assume \( T \longrightarrow T\text{Target} \land T' \)

hence \( \text{TargetTerm } T \longrightarrow (\text{STCal Source } \text{Target})^* (\text{TargetTerm } T') \)

by \((\text{simp add: STCal-steps})\)

ultimately obtain \( P'' \, Q'' \) where \( A2: \text{TargetTerm } T \longrightarrow (\text{STCal Source } \text{Target})^* P'' \)

and \( A3: \text{TargetTerm } T' \longrightarrow (\text{STCal Source } \text{Target})^* Q'' \) and \( A4: (P'', Q'') \in \text{Rel} \)

using \text{corSim}

by blast

from \( A2 \) obtain \( T'' \) where \( A5: T' \longrightarrow T\text{Target} \land T' \land T'' \in T \, P'' \)

by \((\text{auto simp add: STCal-steps})\)

from \( A3 \) obtain \( T'' \) where \( A7: T' \longrightarrow T\text{Target} \land T' \land T'' \in T \, Q'' \)

by \((\text{auto simp add: STCal-steps})\)

from \( A4 \, A6 \, A8 \) trel have \((T'', T'') \in T\text{Rel}^+ \)

by blast

with \( A5 \, A7 \)

show \( \exists T'' \) \( T' \). \( T'' \longrightarrow T\text{Target} \land T'' \land T' \longrightarrow T\text{Target} \land T'' \land (T'', T'') \in T\text{Rel}^+ \)

by blast

next

case \((\text{step } T \, Q \, T \, R)\)

assume \( \land TQ'. \, TQ \longrightarrow T\text{Target} \land T' \longrightarrow \exists T'' \, T'''. \, TQ' \longrightarrow T\text{Target} \land T'' \land T' \longrightarrow T\text{Target} \land T'' \land T'' \land (T'', T'') \in T\text{Rel}^+ \)

moreover assume \((T, T') \in T\text{Rel} \)

hence \( \text{TargetTerm } T \longrightarrow (\text{STCal Source } \text{Target})^* (\text{TargetTerm } T') \)

by \((\text{simp add: STCal-steps})\)

ultimately obtain \( Q'' \, R'' \) where \( B1: \text{TargetTerm } T \longrightarrow (\text{STCal Source } \text{Target})^* Q'' \)

and \( B2: \text{TargetTerm } T' \longrightarrow (\text{STCal Source } \text{Target})^* R'' \) and \( B3: (Q'', R'') \in \text{Rel} \)

using \text{corSim}

by blast

from \( B1 \) obtain \( TQ'' \) where \( B4: TQ'' \in T \, Q'' \) and \( B5: TQ \longrightarrow T\text{Target} \land TQ'' \)

by \((\text{auto simp add: STCal-steps})\)

from \( B2 \) obtain \( TR'' \) where \( B6: TR'' \in T \, R'' \) and \( B7: TR' \longrightarrow T\text{Target} \land TR'' \)

by \((\text{auto simp add: STCal-steps})\)

from \( B3 \, B4 \, B6 \) trel have \((TQ'', TR'') \in T\text{Rel}^+ \)

by simp

with \( B5 \, B7 \)

show \( \exists TQ'' \, TR''. \, TQ \longrightarrow T\text{Target} \land TQ'' \land TR' \longrightarrow T\text{Target} \land TR'' \land (TQ'', TR'') \in T\text{Rel}^+ \)

by blast

qed

moreover have \( \text{trans } (T\text{Rel}^+) \)

by simp

moreover assume \( T \longrightarrow T\text{Target} \land T' \)

ultimately

show \( \exists T'' \, TR''. \, T \longrightarrow T\text{Target} \land T'' \land TR' \longrightarrow T\text{Target} \land T'' \land (T'', TR'') \in T\text{Rel}^+ \)
using A reduction-correspondence-simulation-condition-trans[where Rel=TRel\(^+\)]
and Cal=Target]
by blast
qed
qed ultimately show ?thesis
by simp
qed

lemma (in encoding) indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation:
  fixes TRel :: (\{procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}) set
  assumes cSim: weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)
  shows weak-reduction-correspondence-simulation (TRel\(^+\)) Target
using cSim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel(\{)\[where TRel=TRel]
rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where
Rel=indRelRTPO TRel and TRel=TRel]
by blast

lemma (in encoding) indRelLTPO-impl-TRel-is-weak-reduction-correspondence-simulation:
  fixes TRel :: (\{procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}) set
  assumes cSim: weak-reduction-correspondence-simulation (indRelLTPO TRel) (STCal Source Target)
  shows weak-reduction-correspondence-simulation (TRel\(^+\)) Target
using cSim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(\{)\[where TRel=TRel]
rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where
Rel=indRelLTPO TRel and TRel=TRel]
by blast

lemma (in encoding)
  rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-correspondence-simulation:
  fixes TRel :: (\{procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\}) set
  and Rel :: ((\{procS \times \{\}
  \} procS, \{procT \times \{\}
  \} procT \times \{\}
  \} procS)) Proc \times ((\{procS, \{procT \times \{\}
  \} procT \times \{\}
  \} procT \times \{\})) Proc set
  assumes corSim: weak-reduction-correspondence-simulation-relation (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-relation (TRel\(^+\)) Target
proof
  from corSim target trel have A: weak-reduction-simulation (TRel\(^+\)) Target
using rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-correspondence-simulation[where TRel=TRel
and Rel=Rel]
by blast
moreover have \(\forall P Q Q'. (P, Q) \in TRel^* \land Q \longrightarrow Target^* Q' \longrightarrow
(\exists P'' Q''. P \longrightarrow Target^* P'' \land Q' \longrightarrow Target^* Q'') \land (P'', Q'') \in TRel^+\)
proof clarify
fix TP TQ TQ'
assume \((TP, TQ) \in TRel^* \land TQ \longrightarrow Target^* TQ'\)
thus \(\exists TP'' TQ''. TP \longrightarrow Target^* TP'' \land TQ' \longrightarrow Target^* TQ'' \land (TP'', TQ'') \in TRel^+\)
proof (induct arbitrary: TQ')
  fix TQ'
  assume TP \longrightarrow Target^* TQ'
moreover have TQ' \longrightarrow Target^* TQ'
    by (simp add: steps-refl)
moreover have \((TQ', TQ') \in TRel^*\)
    by simp
ultimately show \(\exists TP'' TQ''. TP \longrightarrow Target^* TP'' \land TQ' \longrightarrow Target^* TQ'' \land (TP'', TQ'') \in TRel^*\)
    by blast
next
case (step TQ TR TR')
assume \(\land TQ'. TQ \longrightarrow Target^* TQ' \longrightarrow \exists TP'' TQ''. TP \longrightarrow Target^* TP'' \land TQ' \longrightarrow Target^* TQ'' \land (TP'', TQ'') \in TRel^*\)
moreover assume \((TQ', TR') \in TRel\)
with corSim have \(\land TR'. TR \longrightarrow Target^* TR' \longrightarrow \exists TQ'' TR''. TQ \longrightarrow Target^* TQ''\)
\[ \land TR' \longrightarrow Target^* TR'' \land (TQ'', TR'') \in TRel^* \]

**proof clarify**

fix \( TR' \)

assume \((TQ, TR) \in TRel\)

with target have \((TargetTerm TQ, TargetTerm TR) \in Rel\)

by simp

moreover assume \( TR \longrightarrow Target^* TR' \)

hence \( TargetTerm TR \longrightarrow (STCal Source Target)^* (TargetTerm TR') \)

by \((simp add: STCal-steps)\)

ultimately obtain \( Q'' \land R'' \) where \( B1: TargetTerm TQ \longrightarrow (STCal Source Target)^* Q'' \)

and \( B2: TargetTerm TR' \longrightarrow (STCal Source Target)^* R'' \) and \( B3: (Q'', R'') \in Rel \)

using \( corSim \)

by \( blast \)

from \( B1 \) obtain \( TQ'' \) where \( B4: TQ'' \in T Q'' \) and \( B5: TQ \longrightarrow Target^* TQ'' \)

by \((auto simp add: STCal-steps)\)

from \( B2 \) obtain \( TR'' \) where \( B6: TR'' \in T R'' \) and \( B7: TR' \longrightarrow Target^* TR'' \)

by \((auto simp add: STCal-steps)\)

from \( B3 B4 B6 trrel \) have \((TQ'', TR'') \in TRel^* \)

by simp

with \( B5 B7 \)

show \( \exists TQ'' TR''. TQ \longrightarrow Target^* TQ'' \land TR' \longrightarrow Target^* TR'' \land (TQ'', TR'') \in TRel^* \)

by \( blast \)

qed

moreover assume \( TR \longrightarrow Target^* TR' \)

moreover have \( Trans (TRel^*) \)

using \( trans-trancl[of TRel] \)

by simp

ultimately show \( \exists TP'' TR''. TP \longrightarrow Target^* TP'' \land TR' \longrightarrow Target^* TR'' \land (TP'', TR'') \in TRel^* \)

using \( A \) reduction-correspondence-simulation-condition-trans[where \( Rel=TRel^* \)

and \( Cal=TTarget \)]

by \( blast \)

qed

ultimately show \(?thesis\)

by simp

qed

**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 \( weak-reduction-correspondence-simulation (TRel^*) Target \)

using \( corSim \) \( indRelTEQ.target[where \( TRel=TRel\)] \) \( indRelTEQ-to-TRel(4)[where \( TRel=TRel\)] \)

trans-closure-of-TRel-refl-cond

rel-with-target-impl-refC-transC-TRel-is-weak-reduction-correspondence-simulation[

where \( Rel=indRelTEQ TRel \) and \( TRel=TRel\)]

by \( blast \)

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

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

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

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

proof

from \( corSim \) show \( weak-reduction-correspondence-simulation (TRel^+) (Calculus TWB) \)

using \( indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation[where \( TRel=TRel\)] \)

by \((simp add: STCalWB-def calS calT)\)

next

from \( corSim \) show \( rel-weakly-respects-barbs (TRel^+) TWB \)

using \( indRelRTPO-impl-TRel-weakly-respects-barbs[where \( TRel=TRel\)] \)

weak-respection-of-barbs-and-closures(3)[where \( Rel=TRel \) and \( CWB=TWB\)]

by \( blast \)

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

*Proof*

- **fixes** \(\text{TRel} : (\text{procT} \times \text{procT}) \text{ set}
- **assumes** \(\text{corSim} := \text{weak-barbed-correspondence-simulation (indRelLTPO TRel) (STCalWB SWB TWB)}
- **shows** \(\text{weak-barbed-correspondence-simulation (TRel')} \text{ TWB}

**next**

- **from** \(\text{corSim} \text{ show weak-reduction-correspondence-simulation (TRel')} \text{ (Calculus TWB)}
- **using** \(\text{indRelLTPO-impl-TRel-is-weak-reduction-correspondence-simulation[where TRel=TRel]}
- **by** \((\text{simp add: STCalWB-def calS calT)}

**qed**

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

**next**

- **from** \(\text{corSim} \text{ show weak-reduction-correspondence-simulation (TRel')} \text{ (Calculus TWB)}
- **using** \(\text{indRelTEQ-impl-TRel-is-weak-reduction-correspondence-simulation[where TRel=TRel]}
- **by** \((\text{simp add: STCalWB-def calS calT)}

**qed**

If \(\text{indRelRTPo, indRelLTPO, or indRelTEQ}\) is a bisimulation then so is the corresponding target term relation.

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

**proof**

- **from** \(\text{bisim} \text{ target trel show weak-reduction-simulation (TRel')} \text{ Target}
- **using** \(\text{rel-with-target-impl-transC-TRel-is-weak-reduction-simulation[where TRel=TRel} \text{ and Rel=Rel]}\n  **by** \(\text{blast}

**next**

- **show \(\forall \text{P Q Q'}. (P, Q) \in TRel' \land Q \rightarrow Target* Q' \rightarrow (\exists P'. P \rightarrow Target* P' \land (P', Q') \in TRel')

**proof**

- **clarify**
- **fix** \(\text{TP TQ TQ'}\)
  - **assume** \((\text{TP, TQ}) \in TRel' \text{ and TQ} \rightarrow Target* TQ'\)
  - **thus** \(\exists \text{TP', TP} \rightarrow Target* \text{TP' \land (TP', TQ') \in TRel'}\)
  **proof**

  - **(induct arbitrary: TQ')
  - **fix** \(\text{TQ TQ'}\)
  - **assume** \((\text{TP, TQ}) \in TRel\)
  - **with** \(\text{target have (TargetTerm TP, TargetTerm TQ) \in Rel}\)
  **by** \(\text{simp}
  - **moreover assume** \(\text{TQ} \rightarrow Target* TQ'\)
  - **hence** \(\text{TargetTerm TQ} \rightarrow (\text{STCal Source Target)* (TargetTerm TQ')}\)
  **by** \((\text{simp add: STCal-steps})\)
  - **ultimately obtain** \(\text{P' where A2: TargetTerm TP \rightarrow (STCal Source Target)P'}\)
and A3: \( (P', \text{TargetTerm } TQ') \in \text{Rel} \)

using bisim
by blast

from A2 obtain TP' where A4: TP \rightarrow Target* TP' and A5: TP' \in T P'
by (auto simp add: STCal-steps)

from A3 A5 trel have (TP', TQ') \in TRel+
by simp

with A4 show \( \exists TP'. TP \rightarrow Target* TP' \land (TP', TQ') \in TRel^+ \)
by blast

next
case (step TQ TR TR')
assume (TQ, TR) \in TRel
with target have (TargetTerm TQ, TargetTerm TR) \in Rel
by simp

moreover assume TR \rightarrow Target* TR'

hence TargetTerm TR \rightarrow (STCal Source Target)* (TargetTerm TR')
by (simp add: STCal-steps)

ultimately obtain Q' where B1: TargetTerm TQ \rightarrow (STCal Source Target)* Q'
and B2: (Q', TargetTerm TR') \in Rel

using bisim
by blast

from B1 obtain TQ' where B3: TQ' \in T Q' and B4: TQ \rightarrow Target* TQ'
by (auto simp add: STCal-steps)

assume \( \land TQ', TQ \rightarrow Target* TQ' \rightarrow \exists TP'. TP \rightarrow Target* TP' \land (TP', TQ') \in TRel^+ \)

with B4 obtain TP' where B5: TP \rightarrow Target* TP' and B6: (TP', TQ') \in TRel^+
by blast

from B2 B3 trel have (TQ', TR') \in TRel+
by simp

with B6 have (TP', TR') \in TRel+
by simp

with B5 show \( \exists TP'. TP \rightarrow Target* TP' \land (TP', TR') \in TRel^+ \)
by blast

qed

qed

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

using bisim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel(\_)[where TRel=TRel] rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation[where Rel=indRelRTPO TRel and TRel=TRel]

by blast

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

using bisim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel(\_)[where TRel=TRel] rel-with-target-impl-transC-TRel-is-weak-reduction-bisimulation[where Rel=indRelLTPO TRel and TRel=TRel]

by blast

lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation:
fixes TRel :: ('procT × 'procT) set
and Rel :: ('procS, 'procT) Proc × ('procS, 'procT) Proc) set
assumes bisim: weak-reduction-bisimulation Rel (STCal Source Target)
and target: \( \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \)
and trel: \( \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in TRel^* \)
shows weak-reduction-bisimulation \( (TRel^*) \) Target
proof
from bisim target trel show weak-reduction-simulation \((T\rel^*)\) Target
  using rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-simulation\[\text{where } T\rel=T\rel\]
  and Rel=Rel]
  by blast
next
show \(\forall P\ Q\ Q'.\ (P,\ Q) \in T\rel^* \land Q \longrightarrow \text{Target}\ast\ Q' \longrightarrow (\exists P'.\ P \longrightarrow \text{Target}\ast\ P' \land (P',\ Q') \in T\rel^*)\)
proof clarify
fix TP TQ TQ'
assume \((TP,\ TQ) \in T\rel^* \land TQ \longrightarrow \text{Target}\ast\ TQ'\)
thus \(\exists TP'.\ TP \longrightarrow \text{Target}\ast\ TP' \land (TP',\ TQ') \in T\rel^*\)
proof (induct arbitrary: \(TQ'\))
fix TQ'
assume TP \longrightarrow \text{Target}\ast\ TQ'
moreover have \((TQ',\ TQ') \in T\rel^*\)
  by simp
ultimately show \(\exists TP'.\ TP \longrightarrow \text{Target}\ast\ TP' \land (TP',\ TQ') \in T\rel^*\)
  by blast
next
case (step TQ TR TR')
assume \((TQ,\ TR) \in T\rel\)
with target have \((\text{TargetTerm} TQ,\ \text{TargetTerm} TR) \in \text{Rel}\)
  by simp
moreover assume TR \longrightarrow \text{Target}\ast\ TR'
then \text{TargetTerm} TR \longrightarrow (\text{STCal Source Target}) \ast (\text{TargetTerm} TR')
  by (simp add: STCal-steps)
ultimately obtain Q' where B1: \(\text{TargetTerm} TQ \longrightarrow (\text{STCal Source Target}) \ast Q'\)
  and B2: \((Q',\ \text{TargetTerm} TR') \in \text{Rel}\)
  using bisim
  by blast
from B1 obtain TQ' where B3: \(TQ' \in T\ Q' \land B4: TQ \longrightarrow \text{Target}\ast\ TQ'\)
  by (auto simp add: STCal-steps)
assume \(\land TQ'.\ TQ \longrightarrow \text{Target}\ast\ TQ' \longrightarrow \exists TP'.\ TP \longrightarrow \text{Target}\ast\ TP' \land (TP',\ TQ') \in T\rel^*\)
with B4 obtain TP' where B5: \(TP \longrightarrow \text{Target}\ast\ TP' \land B6: (TP',\ TQ') \in T\rel^*\)
  by blast
from B2 B3 trel have \((TQ',\ TR') \in T\rel^*\)
  by simp
with B6 have \((TP',\ TR') \in T\rel^*\)
  by simp
with B5 show \(\exists TP'.\ TP \longrightarrow \text{Target}\ast\ TP' \land (TP',\ TR') \in T\rel^*\)
  by blast
qed
qed

lemma \[\text{in encoding}\] indRelTEQ-impl-TRel-is-weak-reduction-bisimulation:
  fixes TRel :: ('procT \times 'procT) set
  assumes bisim: weak-reduction-bisimulation \((\text{indRelTEQ} T\rel)\) \((\text{STCal Source Target})\)
  shows weak-reduction-bisimulation \((T\rel')\) Target
    using bisim indRelTEQ.target[\text{where } T\rel=T\rel]\ indRelTEQ-to-TRel(4)[\text{where } T\rel=T\rel]
    trans-closure-of-Trel-refl-cond
    rel-with-target-impl-reflC-transC-TRel-is-weak-reduction-bisimulation[\text{where } Rel=\text{indRelTEQ} T\rel \land T\rel=\text{TRel}]\]
    by blast

lemma \[\text{in encoding}\] rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: ('procT \times 'procT) set
  and Rel :: ('procS, 'procT) Proc \times ('procS, 'procT) Proc set
  assumes bisim: strong-reduction-bisimulation Rel \((\text{STCal Source Target})\)
  and target: \(\forall T1\ T2.\ (T1,\ T2) \in T\rel \longrightarrow \text{(TargetTerm} T1,\ \text{TargetTerm} T2) \in \text{Rel}\)
  and trel: \(\forall T1\ T2.\ \text{(TargetTerm} T1,\ \text{TargetTerm} T2) \in \text{Rel} \longrightarrow (T1,\ T2) \in T\rel^+\)
shows strong-reduction-bisimulation \((T\text{Rel}^+)\) Target

proof

from bisim target trel show strong-reduction-simulation \((T\text{Rel}^+)\) Target

using rel-with-target-impl-transC-T\text{Rel}-is-strong-reduction-simulation[where Rel=Rel and T\text{Rel}=T\text{Rel}]

by blast

next

show \(\forall P\ Q\ Q'.\ (P,\ Q)\in T\text{Rel}^+\land Q\mapsto\text{Target }Q'\longrightarrow (\exists P'.\ P\mapsto\text{Target }P'\land (P',\ Q')\in T\text{Rel}^+)

proof clarify

fix \(TP\ TQ\ TQ'

assume \((TP',\ TQ)\in T\text{Rel}^+ \text{ and } TQ\mapsto\text{Target }TQ'

thus \(\exists TP'.\ TP\mapsto\text{Target }TP'\land (TP',\ TQ')\in T\text{Rel}^+

proof (induct arbitrary: \(TQ'

fix \(TQ\ TQ'

assume \((TP',\ TQ)\in T\text{Rel}

with target have \((\text{TargetTerm }TP,\ \text{TargetTerm }TQ)\in \text{Rel}

by simp

moreover assume \(TQ\mapsto\text{Target }TQ'

hence \(\text{TargetTerm }TQ\mapsto(\text{STCal Source Target})\ (\text{TargetTerm }TQ')

by (simp add: STCal-step)

ultimately obtain \(P'\) where \(A2:\ \text{TargetTerm }TP\mapsto(\text{STCal Source Target})\ P'

and \(A3:\ (P',\ \text{TargetTerm }TQ')\in \text{Rel}

using bisim

by blast

from \(A2\) obtain \(TP'\) where \(A4:\ TP\mapsto\text{Target }TP'\land A5:\ TP'\in T\ P'

by (auto simp add: STCal-step)

from \(A3\ A5\) trel have \((TP',\ TQ')\in T\text{Rel}^+

by simp

with \(A4\) show \(\exists TP'.\ TP\mapsto\text{Target }TP'\land (TP',\ TQ')\in T\text{Rel}^+

by blast

next

case (step \(TQ\ TR\ TR')

assume \((TQ,\ TR)\in T\text{Rel}

with target have \((\text{TargetTerm }TQ,\ \text{TargetTerm }TR)\in \text{Rel}

by simp

moreover assume \(TR\mapsto\text{Target }TR'

hence \(\text{TargetTerm }TR\mapsto(\text{STCal Source Target})\ (\text{TargetTerm }TR')

by (simp add: STCal-step)

ultimately obtain \(Q'\) where \(B1:\ \text{TargetTerm }TQ\mapsto(\text{STCal Source Target})\ Q'

and \(B2:\ (Q',\ \text{TargetTerm }TR')\in \text{Rel}

using bisim

by blast

from \(B1\) obtain \(TQ'\) where \(B3:\ TQ'\in T\ Q'\land B4:\ TQ\mapsto\text{Target }TQ'

by (auto simp add: STCal-step)

assume \(\land TQ'.\ TQ\mapsto\text{Target }TQ'\longrightarrow \exists TP'.\ TP\mapsto\text{Target }TP'\land (TP',\ TQ')\in T\text{Rel}^+

with \(B4\) obtain \(TP'\) where \(B5:\ TP\mapsto\text{Target }TP'\land B6:\ (TP',\ TQ')\in T\text{Rel}^+

by blast

from \(B2\ B3\) trel have \((TQ',\ TR')\in T\text{Rel}^+

by simp

with \(B6\) have \((TP',\ TR')\in T\text{Rel}^+

by simp

with \(B5\) show \(\exists TP'.\ TP\mapsto\text{Target }TP'\land (TP',\ TR')\in T\text{Rel}^+

by blast

qed

qed

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

fixes \(T\text{Rel} :: ('procT\times'procT)\) set

assumes bisim: strong-reduction-bisimulation \((\text{indRelRTPO}\ T\text{Rel})\ \text{(STCal Source Target)}

shows strong-reduction-bisimulation \((T\text{Rel}^+)\) Target
using bsim indRelRTPO.target[where TRel=TRel] indRelRTPO-to-TRel[\langle \rangle][where TRel=TRel]
rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation[where
Rel=indRelRTPO TRel and TRel=TRel]

by blast

lemma (in encoding) indRelLTPO-impl-TRel-is-strong-reduction-bisimulation:
fixes TRel :: (\langle \langle proc \rangle \times \langle proc \rangle \rangle) set
assumes bsim: strong-reduction-bisimulation (indRelLTPO TRel) (STCal Source Target)
shows strong-reduction-bisimulation (TRel\langle \rangle\rangle) Target
using bsim indRelLTPO.target[where TRel=TRel] indRelLTPO-to-TRel[\langle \rangle][where TRel=TRel]
rel-with-target-impl-transC-TRel-is-strong-reduction-bisimulation[where
Rel=indRelLTPO TRel and TRel=TRel]
by blast

lemma (in encoding) rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation:
fixes TRel :: (\langle \langle proc \rangle \times \langle proc \rangle \rangle) set
and Rel :: (\langle \langle proc \rangle, \langle proc \rangle \rangle Proc \times (\langle \langle proc \rangle, \langle proc \rangle \rangle Proc) set
assumes bsim: strong-reduction-bisimulation Rel (STCal Source Target)
and target: \forall T1 T2. (T1, T2) \in TRel \to (TargetTerm T1, TargetTerm T2) \in Rel
and trel: \forall T1 T2. (TargetTerm T1, TargetTerm T2) \in Rel \to (T1, T2) \in TRel\langle \rangle\rangle
shows strong-reduction-bisimulation (TRel\langle \rangle\rangle\rangle Target

from bsim target trel show strong-reduction-simulation (TRel\langle \rangle\rangle\rangle Target
using rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-simulation[where Rel=Rel
and TRel=TRel]
by blast

next
show \forall P Q Q'. (P, Q) \in TRel\langle \rangle\rangle \land Q \to Target Q' \to (\exists P'. P \to Target P' \land (P', Q') \in TRel\langle \rangle\rangle\rangle

proof clarify
fix TP TQ TQ'
assume (TP, TQ) \in TRel\langle \rangle\rangle \land TQ \to Target TQ'
thus \exists TP'. TP \to Target TP' \land (TP', TQ') \in TRel\langle \rangle\rangle

proof (induct arbitrary: TQ')
fix TQ'
assume TP \to Target TQ'
thus \exists TP'. TP \to Target TP' \land (TP', TQ') \in TRel\langle \rangle\rangle

by blast

next

case (step TQ TR TR')
assume (TQ, TR) \in TRel
with target have (TargetTerm TQ, TargetTerm TR) \in Rel
by simp
moreover assume TR \to Target TR'
hence TargetTerm TR \to (STCal Source Target) (TargetTerm TR')
by (simp add: STCal-step)
ultimately obtain Q' where B1: TargetTerm TQ \to (STCal Source Target) Q'
and B2: (Q', TargetTerm TR') \in Rel

using bsim

by blast

from B1 obtain TQ' where B3: TQ' \in T Q' and B4: TQ \to Target TQ'
by (auto simp add: STCal-step)
assume \land TQ'. TQ \to Target TQ' \to \exists TP'. TP \to Target TP' \land (TP', TQ') \in TRel\langle \rangle\rangle
with B4 obtain TP' where B5: TP \to Target TP' and B6: (TP', TQ') \in TRel\langle \rangle\rangle

by blast
from B2 B3 trel have (TQ', TR') \in TRel\langle \rangle\rangle
by simp
with B6 have (TP', TR') \in TRel\langle \rangle\rangle
by simp
with B5 show \exists TP'. TP \to Target TP' \land (TP', TR') \in TRel\langle \rangle\rangle

by blast
qed
lemma (in encoding) indRelTEQ-impl-TRel-is-strong-reduction-bisimulation:
  fixes TRel :: (′procT × ′procT) set
  assumes bisim: strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
  shows strong-reduction-bisimulation (TRel *) Target
    using bisim indRelTEQ.target[where TRel=TRel] indRelTEQ-to-TRel(4)[where TRel=TRel]
    trans-closure-of-Trel-refl-cond
    rel-with-target-impl-reflC-transC-TRel-is-strong-reduction-bisimulation[where Rel=indRelTEQ TRel and TRel=TRel]
    by blast

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: (′procT × ′procT) set
  assumes bisim: weak-barbed-bisimulation (indRelRTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-bisimulation (TRel +) TWB
proof
  from bisim show weak-reduction-bisimulation (TRel +) (Calculus TWB)
    using indRelRTPO-impl-TRel-is-weak-reduction-bisimulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from bisim show rel-weakly-respects-barbs (TRel +) TWB
    using indRelRTPO-impl-TRel-weakly-respects-barbs[where TRel=TRel]
    weak-respection-of-barbs-and-closures(3)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelLTPO-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: (′procT × ′procT) set
  assumes bisim: weak-barbed-bisimulation (indRelLTPO TRel) (STCalWB SWB TWB)
  shows weak-barbed-bisimulation (TRel +) TWB
proof
  from bisim show weak-reduction-bisimulation (TRel +) (Calculus TWB)
    using indRelLTPO-impl-TRel-is-weak-reduction-bisimulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from bisim show rel-weakly-respects-barbs (TRel +) TWB
    using indRelLTPO-impl-TRel-weakly-respects-barbs[where TRel=TRel]
    weak-respection-of-barbs-and-closures(3)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelTEQ-impl-TRel-is-weak-barbed-bisimulation:
  fixes TRel :: (′procT × ′procT) set
  assumes bisim: weak-barbed-bisimulation (indRelTEQ TRel) (STCalWB SWB TWB)
  shows weak-barbed-bisimulation (TRel *) TWB
proof
  from bisim show weak-reduction-bisimulation (TRel *) (Calculus TWB)
    using indRelTEQ-impl-TRel-is-weak-reduction-bisimulation[where TRel=TRel]
    by (simp add: STCalWB-def calS calT)
next
  from bisim show rel-weakly-respects-barbs (TRel *) TWB
    using indRelTEQ-impl-TRel-weakly-respects-barbs[where TRel=TRel]
    weak-respection-of-barbs-and-closures(5)[where Rel=TRel and CWB=TWB]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelRTPO-impl-TRel-is-strong-barbed-bisimulation:
  fixes TRel :: (′procT × ′procT) set
  assumes bisim: strong-barbed-bisimulation (indRelRTPO TRel) (STCalWB SWB TWB)
shows strong-barbed-bisimulation \((TRel^+)\) \(TWB\)

proof
from bisim show strong-reduction-bisimulation \((TRel^+)\) \(Calculus TWB\)
using \(\text{indRelRTPO-impl-TRel-is-strong-reduction-bisimulation}[\text{where } TRel=TRel]\)
by \((\text{simp add: } \text{STCalWB-def calS calT})\)

next
from bisim show rel-respects-bars \((TRel^+)\) \(TWB\)
using \(\text{indRelRTPO-impl-TRel-respects-bars}[\text{where } TRel=TRel]\)
respection-of-bars-and-closures(3)[\text{where } Rel=TRel \text{ and } CBWB=TWB]
by blast

qed

lemma \(\text{in } encoding-wrt-bars) \text{ indRelLTPO-impl-TRel-is-strong-barbed-bisimulation:}\nfixes \(TRel :: ('procT \times 'procT) set\)
assumes bisim: \(\text{strong-barbed-bisimulation} \ (\text{indRelLTPO TRel}) \ (\text{STCalWB SWB TWB})\)
shows \(\text{strong-barbed-bisimulation} \ (TRel^+) \ TWB\)

proof
from bisim refl show strong-reduction-bisimulation \((TRel^+)\) \(Calculus TWB\)
using \(\text{indRelLTPO-impl-TRel-is-strong-reduction-bisimulation}[\text{where } TRel=TRel]\)
by \((\text{simp add: } \text{STCalWB-def calS calT})\)

next
from bisim show rel-respects-bars \((TRel^+)\) \(TWB\)
using \(\text{indRelLTPO-impl-TRel-respects-bars}[\text{where } TRel=TRel]\)
respection-of-bars-and-closures(5)[\text{where } Rel=TRel \text{ and } CBWB=TWB]
by blast

qed

lemma \(\text{in } encoding-wrt-bars) \text{ indRelTEQ-impl-TRel-is-strong-barbed-bisimulation:}\nfixes \(TRel :: ('procT \times 'procT) set\)
assumes bisim: \(\text{strong-barbed-bisimulation} \ (\text{indRelTEQ TRel}) \ (\text{STCalWB SWB TWB})\)
shows \(\text{strong-barbed-bisimulation} \ (TRel^+) \ TWB\)

proof
from bisim refl show strong-reduction-bisimulation \((TRel^+)\) \(Calculus TWB\)
using \(\text{indRelTEQ-impl-TRel-is-strong-reduction-bisimulation}[\text{where } TRel=TRel]\)
by \((\text{simp add: } \text{STCalWB-def calS calT})\)

next
from bisim show rel-respects-bars \((TRel^+)\) \(TWB\)
using \(\text{indRelTEQ-impl-TRel-respects-bars}[\text{where } TRel=TRel]\)
respection-of-bars-and-closures(5)[\text{where } Rel=TRel \text{ and } CBWB=TWB]
by blast

qed

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

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

\textbf{inductive-set} \(\text{in encoding)} \text{ indRelRST}\n:: ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set
⇒ ((('procS, 'procT) Proc) \times (('procS, 'procT) Proc)) set
for \(\text{SRel :: ('procS \times 'procS)} set\)
and \(\text{TRel :: ('procT \times 'procT) set}\)
where
\(\text{encR: (SourceTerm } S, \text{ TargetTerm } ([T])) \in \text{indRelRST SRel TRel }\)
\(\text{source: (S1, S2) \in SRel } \Rightarrow (\text{SourceTerm } S1, \text{ SourceTerm } S2) \in \text{indRelRST SRel TRel }\)
\(\text{target: (T1, T2) \in TRel } \Rightarrow (\text{TargetTerm } T1, \text{ TargetTerm } T2) \in \text{indRelRST SRel TRel }\)

\textbf{abbreviation} \(\text{in encoding)} \text{ indRelRST}\text{Infix}\n:: ('procS, 'procT) Proc \Rightarrow ('procS \times 'procS) set \Rightarrow ('procT \times 'procT) set
\[
\Rightarrow (\text{proc}S, \text{proc}T) \text{ Proc} \Rightarrow \text{bool} (- R[<,>] - 75, 75, 75, 75) 80
\]

where
\[
P R[<,>] SRel, TRel > Q \equiv (P, Q) \in \text{indRelRST} SRel TRel
\]

**inductive-set** *(in encoding)* 
\[
\text{indRelRSTPO}
\]
\[
\begin{array}{l}
:: (\text{procS} \times \text{procS}) \text{ set} \Rightarrow (\text{proc}T \times \text{proc}T) \text{ set} \\
\Rightarrow (((\text{procS}, \text{proc}T) \text{ Proc}) \times ((\text{procS}, \text{proc}T) \text{ Proc})) \text{ set}
\end{array}
\]

for 
\[
SRel :: (\text{procS} \times \text{procS}) \text{ set}
\]
and 
\[
TRel :: (\text{proc}T \times \text{proc}T) \text{ set}
\]

where
\[
enR :: (\text{SourceTerm S}, \text{TargetTerm ([S])}) \in \text{indRelRSTPO} SRel TRel |
source :: (S1, S2) \in SRel \Rightarrow (\text{SourceTerm S1}, \text{SourceTerm S2}) \in \text{indRelRSTPO} SRel TRel |
\]
target :: (T1, T2) \in TRel \Rightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{indRelRSTPO} SRel TRel |
\]
\[
\begin{array}{l}
\Rightarrow (P, Q) \in \text{indRelRSTPO} SRel TRel; (Q, R) \in \text{indRelRSTPO} SRel TRel\]
\end{array}
\]

**abbreviation** *(in encoding)* 
\[
\text{indRelRSTPOinfix} ::
\]
\[
\begin{array}{l}
(P, Q) \in \text{indRelRSTPO} SRel TRel\]
\end{array}
\]

where
\[
P R[<,>] SRel, TRel > Q \equiv (P, Q) \in \text{indRelRSTPO} SRel TRel
\]

**lemma** *(in encoding)* 
\[
\text{indRelRSTPO-refl}:
\]
\[
\text{fixes} SRel :: (\text{procS} \times \text{procS}) \text{ set}
\]
and 
\[
\text{assumes} \text{reflS} :: \text{refl} SRel
\]
and 
\[
\text{assumes} \text{reflT} :: \text{refl} TRel
\]
shows 
\[
\text{refl} (\text{indRelRSTPO} SRel TRel)
\]

**proof** 
\[
\text{auto}
\]

fix 
\[
P
\]
show 
\[
P R[<,>] SRel, TRel > P
\]

proof *(cases P)*

\[
\begin{array}{l}
\text{case} (\text{SourceTerm SP}) \\
\text{assume} SP \in S P
\end{array}
\]
\[
\text{with} \text{reflS} \text{ show} P R[<,>] SRel, TRel > P
\]

\[
\text{unfolding} \text{refl-on-def}
\]
\[
\text{by} (\text{simp add: indRelRSTPO.source})
\]

next
\[
\begin{array}{l}
\text{case} (\text{TargetTerm TP}) \\
\text{assume} TP \in T P
\end{array}
\]
\[
\text{with} \text{reflT} \text{ show} P R[<,>] SRel, TRel > P
\]

\[
\text{unfolding} \text{refl-on-def}
\]
\[
\text{by} (\text{simp add: indRelRSTPO.target})
\]

qed

qed

**lemma** *(in encoding)* 
\[
\text{indRelRSTPO-trans}:
\]
\[
\text{fixes} SRel :: (\text{procS} \times \text{procS}) \text{ set}
\]
and 
\[
\text{shows} \text{trans} (\text{indRelRSTPO} SRel TRel)
\]

**proof** 
\[
\text{clarify}
\]

fix 
\[
P Q R
\]
assume 
\[
P R[<,>] SRel, TRel > Q \text{ and} Q R[<,>] SRel, TRel > R
\]
thus 
\[
P R[<,>] SRel, TRel > R
\]

by *(rule indRelRSTPO.trans)*

qed

**lemma** *(in encoding)* 
\[
\text{refl-trans-closure-of-indRelRST}:
\]
\[
\text{fixes} SRel :: (\text{procS} \times \text{procS}) \text{ set}
\]
and \( TRel :: \{\text{'procT} \times \text{'procT}\} \) set

assumes refS: \( \text{refl SRel} \)

and refT: \( \text{refl TRel} \)

shows \( \text{indRelRSTPO SRel TRel} = (\text{indRelRST SRel TRel})^* \)

proof auto

fix \( P \)

assume \( P \preceq R SRel, TRel > Q \)

thus \( (P, Q) \in (\text{indRelRST SRel TRel})^* \)

proof induct

case (encR S)

show \((\text{SourceTerm S}, \text{TargetTerm } [S]) \in (\text{indRelRST SRel TRel})^* \)

using \( \text{indRelRST. encR}[S SRel TRel] \)

by simp

defined

case (source S1 S2)

assume \((S1, S2) \in SRel \)

thus \((\text{SourceTerm S1}, \text{SourceTerm S2}) \in (\text{indRelRST SRel TRel})^* \)

using \( \text{indRelRST. source}[S1 S2 SRel TRel] \)

by simp

defined

case (target T1 T2)

assume \((T1, T2) \in TRel \)

thus \((\text{TargetTerm T1}, \text{TargetTerm T2}) \in (\text{indRelRST SRel TRel})^* \)

using \( \text{indRelRST. target}[T1 T2 TRel SRel] \)

by simp

defined

case (trans P Q R)

assume \((P, Q) \in (\text{indRelRST SRel TRel})^* \) and \((Q, R) \in (\text{indRelRST SRel TRel})^* \)

thus \((P, R) \in (\text{indRelRST SRel TRel})^* \)

by simp

qed defined

next

fix \( P \)

assume \((P, Q) \in (\text{indRelRST SRel TRel})^* \)

thus \( P \preceq R SRel, TRel > Q \)

proof induct

from \( \text{refS} \) \( \text{refT} \) show \( P \preceq R SRel, TRel > P \)

using \( \text{indRelRSTPO-refl}[S SRel TRel] \)

unfolding \( \text{refl-on-def} \)

by simp

defined

next

assume \((P, Q) \in (\text{indRelRST SRel TRel})^* \)

moreover assume \( Q \preceq R SRel, TRel > R \)

hence \( Q \preceq R SRel, TRel > R \)

by (induct, simp-all add: \( \text{indRelRSTPO.intros} \))

ultimately show \( P \preceq R SRel, TRel > R \)

by (rule \( \text{indRelRSTPO.trans} \))

qed

qed

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

\[ \text{:: ('}\text{procS} \times \text{'procS}') \text{ set} \]

\[ \Rightarrow (\text{'}\text{procS} \times \text{'procS}') \text{ set} \]

\[ \Rightarrow ((\text{'}\text{procS} \times \text{'procT}') \text{ Proc} \times ((\text{'}\text{procS} \times \text{'procT}') \text{ Proc}) \text{ set} \]

for \( \text{SRel} :: (\text{'}\text{procS} \times \text{'procS}') \text{ set} \)

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

where

\[ \text{encL} :: (\text{TargetTerm } [S], \text{SourceTerm S}) \in \text{indRelLST SRel TRel} \mid \]

\[ \text{source} :: (S1, S2) \in SRel \Rightarrow (\text{SourceTerm S1, SourceTerm S2}) \in \text{indRelLST SRel TRel} \mid \]

\[ \text{target} :: (T1, T2) \in TRel \Rightarrow (\text{TargetTerm T1, TargetTerm T2}) \in \text{indRelLST SRel TRel} \]
abbreviation (in encoding) indRelLSTinfix

\[
\begin{align*}
\text{fixes } & SRel :: (\text{procS} \times \text{procS}) \Rightarrow (\text{procT} \times \text{procT}) \Rightarrow (\text{procS} \times \text{procT}) \Rightarrow \text{bool} \\
\text{where } & P \ R \ SRel \ Q \equiv (P, Q) \in \text{indRelLST} SRel TRel
\end{align*}
\]

inductive-set (in encoding) indRelLSTPO

\[
\begin{align*}
\text{fixes } & SRel :: (\text{procS} \times \text{procS}) \Rightarrow (\text{procT} \times \text{procT}) \Rightarrow (\text{procS} \times \text{procT}) \Rightarrow \text{bool} \\
\text{where } & P \ SRel \ Q \equiv (P, Q) \in \text{indRelLST} SRel TRel
\end{align*}
\]

abbreviation (in encoding) indRelLSTPoInfix

\[
\begin{align*}
\text{fixes } & SRel :: (\text{procS} \times \text{procS}) \Rightarrow (\text{procT} \times \text{procT}) \Rightarrow (\text{procS} \times \text{procT}) \Rightarrow \text{bool} \\
\text{where } & P \ SRel \ Q \equiv (P, Q) \in \text{indRelLST} SRel TRel
\end{align*}
\]

lemma (in encoding) indRelLSTPO-refl:

\[
\begin{align*}
\text{fixes } & SRel :: (\text{procS} \times \text{procS}) \Rightarrow (\text{procT} \times \text{procT}) \Rightarrow (\text{procS} \times \text{procT}) \Rightarrow \text{bool} \\
\text{where } & P \ SRel \ Q \equiv (P, Q) \in \text{indRelLST} SRel TRel
\end{align*}
\]

proof auto

fix P

show \( P \leq [\cdot] L \leq SRel, TRel > P \)

proof (cases P)

case (SourceTerm SP)

assume SP \in S P

with reflS show \( P \leq [\cdot] L \leq SRel, TRel > P \)

unfolding refl-on-def

by (simp add: indRelLSTPO.source)

next

case (TargetTerm TP)

assume TP \in T P

with reflT show \( P \leq [\cdot] L \leq SRel, TRel > P \)

unfolding refl-on-def

by (simp add: indRelLSTPO.target)

qed

qed

lemma (in encoding) indRelLSTPO-trans:

\[
\begin{align*}
\text{fixes } & SRel :: (\text{procS} \times \text{procS}) \Rightarrow (\text{procT} \times \text{procT}) \Rightarrow (\text{procS} \times \text{procT}) \Rightarrow \text{bool} \\
\text{where } & P \ SRel \ Q \equiv (P, Q) \in \text{indRelLST} SRel TRel
\end{align*}
\]

proof clarify

fix P Q R

assume \( P \leq [\cdot] L \leq SRel, TRel > Q \) and \( Q \leq [\cdot] L \leq SRel, TRel > R \)

thus \( P \leq [\cdot] L \leq SRel, TRel > R \)

by (rule indRelLSTPO.trans)

qed
lemma (in encoding) refl-trans-closure-of-indRelLST:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes reflS: refl SRel
  and reflT: refl TRel
  shows indRelLSTPO SRel TRel = (indRelLST SRel TRel)*
proof auto
  fix P Q
  assume P ≲L< SRel, TRel> Q
  thus (P, Q) ∈ (indRelLST SRel TRel)*
proof induct
  case (encL S)
  show (TargetTerm ([S]), SourceTerm S) ∈ (indRelLST SRel TRel)*
    using indRelLST. encL[of S SRel TRel]
    by simp
next
  case (source S1 S2)
  assume (S1, S2) ∈ SRel
  thus (SourceTerm S1, SourceTerm S2) ∈ (indRelLST SRel TRel)*
    using indRelLST. source[of S1 S2 SRel TRel]
    by simp
next
  case (target T1 T2)
  assume (T1, T2) ∈ TRel
  thus (TargetTerm T1, TargetTerm T2) ∈ (indRelLST SRel TRel)*
    using indRelLST. target[of T1 T2 TRel SRel]
    by simp
next
  case (trans P Q R)
  assume (P, Q) ∈ (indRelLST SRel TRel)* and (Q, R) ∈ (indRelLST SRel TRel)*
  thus (P, R) ∈ (indRelLST SRel TRel)*
    by simp
  qed
next
  fix P Q
  assume (P, Q) ∈ (indRelLST SRel TRel)*
  thus P ≲L< SRel, TRel> Q
proof induct
  from reflS reflT show P ≲L< SRel, TRel> P
    using indRelLSTPO-refl[of SRel TRel]
    unfolding refl-on-def
    by simp
next
  case (step Q R)
  assume P ≲L< SRel, TRel> Q
  moreover assume Q R [L< SRel, TRel> R
  hence Q ≲L< SRel, TRel> R
    by (induct, simp-all add: indRelLSTPO.intros)
  ultimately show P ≲L< SRel, TRel> R
    by (rule indRelLSTPO.trans)
  qed
next
inductive-set (in encoding) indRelST
:: ('procS × 'procS) set ⇒ ('procT × 'procT) set
⇒ (((procS, procT) Proc) × ((procS, procT) Proc)) set
for SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
where
  encR: (SourceTerm S, TargetTerm ([S])) ∈ indRelST SRel TRel |
  encL: (TargetTerm ([S]), SourceTerm S) ∈ indRelST SRel TRel |
source: \((S_1, S_2) \in SRel \implies (SourceTerm S_1, SourceTerm S_2) \in \text{indRelST} SRel TRel\) | target: \((T_1, T_2) \in TRel \implies (TargetTerm T_1, TargetTerm T_2) \in \text{indRelST} SRel TRel\)

**abbreviation (in encoding) indRelSTinfix**

\[\langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \langle \text{procS} \times \text{procS} \rangle \text{set} \implies \langle \text{procT} \times \text{procT} \rangle \text{set} \implies \langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \text{bool} \ (\text{R[\[<\cdot,-\cdot<\] [\[<\cdot,-\cdot<\] [\[<\cdot,-\cdot<\] \)}\]

where

\(P \ R[\[\langle SRel, TRel \rangle \ Q \equiv (P, Q) \in \text{indRelST} SRel TRel\]

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

\[\text{fixes} \ SRel : (\text{procS} \times \text{procS}) \text{set} \quad \text{and} \ TRel : (\text{procT} \times \text{procT}) \text{set} \quad \text{assumes} \ \text{symmS} : \text{sym} SRel \quad \text{and} \ \text{symmT} : \text{sym} TRel \quad \text{shows} \ \text{sym} (\text{indRelST} SRel TRel) \quad \text{unfolding} \ \text{sym-def}\]

**proof clarify**

fix \(P \ Q\)

assume \((P, Q) \in \text{indRelST} SRel TRel\)

thus \((Q, P) \in \text{indRelST} SRel TRel\)

**proof induction**

\[\text{case} (\text{encR} S) \]

\[\text{show} \ (\text{TargetTerm} \ \langle [S] \rangle, \text{SourceTerm} \ S) \in \text{indRelST} SRel TRel \quad \text{by} \ \text{(rule indRelST.encL)}\]

next

\[\text{case} (\text{encL} S) \]

\[\text{show} \ (\text{SourceTerm} \ S, \text{TargetTerm} \ \langle [S] \rangle) \in \text{indRelST} SRel TRel \quad \text{by} \ \text{(rule indRelST.encR)}\]

next

\[\text{case} \ (\text{source} S_1 \ S_2) \]

\[\text{assume} \ (S_1, S_2) \in SRel \quad \text{with} \ \text{symmS} \text{ show} \ (\text{SourceTerm} \ S_2, \text{SourceTerm} \ S_1) \in \text{indRelST} SRel TRel \quad \text{unfolding} \ \text{sym-def} \quad \text{by} \ \text{(simp add: indRelST.source)}\]

next

\[\text{case} \ (\text{target} T_1 \ T_2) \]

\[\text{assume} \ (T_1, T_2) \in TRel \quad \text{with} \ \text{symmT} \text{ show} \ (\text{TargetTerm} \ T_2, \text{TargetTerm} \ T_1) \in \text{indRelST} SRel TRel \quad \text{unfolding} \ \text{sym-def} \quad \text{by} \ \text{(simp add: indRelST.target)}\]

**qed**

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

\[\langle \text{procS} \times \text{procS} \rangle \text{set} \implies \langle \text{procT} \times \text{procT} \rangle \text{set} \implies \langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \langle \text{procS} \times \text{procS} \rangle \text{set} \quad \text{and} \ TRel : (\text{procT} \times \text{procT}) \text{set}\]

**abbreviation (in encoding) indRelSTEQinfix**

\[\langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \langle \text{procS} \times \text{procS} \rangle \text{set} \implies \langle \text{procT} \times \text{procT} \rangle \text{set} \implies \langle \text{procS}, \text{procT} \rangle \text{Proc} \implies \text{bool} \ (\text{R[\[<\cdot,-\cdot<\] [\[<\cdot,-\cdot<\] [\[<\cdot,-\cdot<\] \)}\]

where

\(P \sim \ 1 < SRel, TRel > \ Q \equiv (P, Q) \in \text{indRelST} SRel TRel\)
lemma \((\text{in encoding})\) \text{indRelSTEQ-refl}:
\[
\text{fixes } SRel :: (\text{procS} \times \text{procS}) \text{ set } \\
\text{and } TRel :: (\text{procT} \times \text{procT}) \text{ set } \\
\text{assumes } \text{reflT}: \text{refl TRel} \\
\text{shows } \text{refl } (\text{indRelSTEQ } SRel \ TRel) \\
\text{unfolding } \text{refl-on-def} \\
\text{proof auto} \\
\text{fix } P \\
\text{show } P \sim [\cdot] < SRel, TRel > P \\
\text{proof (cases } P \text{)} \\
\text{case } (\text{SourceTerm } SP) \\
\text{assume } SP \in S P \\
\text{moreover have } \text{SourceTerm } SP \sim [\cdot] < SRel, TRel > \text{ TargetTerm } ([SP]) \\
\text{by (rule indRelSTEQ.encR)} \\
\text{moreover have } \text{TargetTerm } ([SP]) \sim [\cdot] < SRel, TRel > \text{ SourceTerm } SP \\
\text{by (rule indRelSTEQ.encL)} \\
\text{ultimately show } P \sim [\cdot] < SRel, TRel > P \\
\text{by (simp add: indRelSTEQ.trans[where } P=\text{SourceTerm } SP \text{ and } Q=\text{TargetTerm } ([SP])])
\text{next} \\
\text{case } (\text{TargetTerm } TP) \\
\text{assume } TP \in T P \\
\text{with } \text{reflT show } P \sim [\cdot] < SRel, TRel > P \\
\text{unfolding } \text{refl-on-def} \\
\text{qed}
\text{qed}
\]

lemma \((\text{in encoding})\) \text{indRelSTEQ-symm}:
\[
\text{fixes } SRel :: (\text{procS} \times \text{procS}) \text{ set } \\
\text{and } TRel :: (\text{procT} \times \text{procT}) \text{ set } \\
\text{assumes } \text{symmS}: \text{sym } SRel \\
\text{and } \text{symmT}: \text{sym } TRel \\
\text{shows } \text{sym } (\text{indRelSTEQ } SRel \ TRel) \\
\text{unfolding } \text{sym-def} \\
\text{proof clarify} \\
\text{fix } P \ Q \\
\text{assume } P \sim [\cdot] < SRel, TRel > P \\
\text{thus } Q \sim [\cdot] < SRel, TRel > Q \\
\text{proof induct} \\
\text{case } (\text{encR } S) \\
\text{show } \text{TargetTerm } ([S]) \sim [\cdot] < SRel, TRel > \text{ SourceTerm } S \\
\text{by (rule indRelSTEQ.encL)} \\
\text{next} \\
\text{case } (\text{encL } S) \\
\text{show } \text{SourceTerm } S \sim [\cdot] < SRel, TRel > \text{ TargetTerm } ([S]) \\
\text{by (rule indRelSTEQ.encR)} \\
\text{next} \\
\text{case } (\text{source } S1 S2) \\
\text{assume } (S1, S2) \in SRel \\
\text{with } \text{symmS show } \text{SourceTerm } S2 \sim [\cdot] < SRel, TRel > \text{ SourceTerm } S1 \\
\text{unfolding } \text{sym-def} \\
\text{by (simp add: indRelSTEQ.source)} \\
\text{next} \\
\text{case } (\text{target } T1 T2) \\
\text{assume } (T1, T2) \in TRel \\
\text{with } \text{symmT show } \text{TargetTerm } T2 \sim [\cdot] < SRel, TRel > \text{ TargetTerm } T1 \\
\text{unfolding } \text{sym-def} \\
\text{by (simp add: indRelSTEQ.target)} \\
\text{next} \\
\text{case } (\text{trans } P \ Q \ R)
assume $R \sim [\cdot] < S_{Rel}, T_{Rel} > Q$ and $Q \sim [\cdot] < S_{Rel}, T_{Rel} > P$

thus $R \sim [\cdot] < S_{Rel}, T_{Rel} > P$

by (rule indRelSTEQ.trans)

qed

lemma (in encoding) indRelSTEQ-trans:

fixes $S_{Rel} :: (\text{'procS} \times \text{'procS})$ set

and $T_{Rel} :: (\text{'procT} \times \text{'procT})$ set

shows $\text{trans} (\text{indRelSTEQ} S_{Rel} T_{Rel})$

unfolding $\text{trans-def}$

proof clarify

fix $P, Q, R$

assume $P \sim [\cdot] < S_{Rel}, T_{Rel} > Q$ and $Q \sim [\cdot] < S_{Rel}, T_{Rel} > R$

thus $P \sim [\cdot] < S_{Rel}, T_{Rel} > R$

by (rule indRelSTEQ.trans)

qed

lemma (in encoding) refl-trans-closure-of-indRelST:

fixes $S_{Rel} :: (\text{'procS} \times \text{'procS})$ set

and $T_{Rel} :: (\text{'procT} \times \text{'procT})$ set

assumes reflT: $\text{refl} T_{Rel}$

shows $\text{indRelSTEQ} S_{Rel} T_{Rel} = (\text{indRelST} S_{Rel} T_{Rel})^*$

proof auto

fix $P, Q$

assume $P \sim [\cdot] < S_{Rel}, T_{Rel} > Q$

thus $(P, Q) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

proof induct

case (encR $S$)

show $(\text{SourceTerm } S, \text{TargetTerm} ([\cdot] S)) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

using $\text{indRelST.encR[of S S_{Rel} T_{Rel}]}$

by simp

next

case (encL $S$)

show $(\text{TargetTerm} ([\cdot] S), \text{SourceTerm} S) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

using $\text{indRelST.encL[of S S_{Rel} T_{Rel}]}$

by simp

next

case (source $S1$ $S2$)

assume $(S1, S2) \in S_{Rel}$

thus $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

using $\text{indRelST.source[of S1 S2 S_{Rel} T_{Rel}]}$

by simp

next

case (target $T1$ $T2$)

assume $(T1, T2) \in T_{Rel}$

thus $(\text{TargetTerm } T1, \text{TargetTerm } T2) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

using $\text{indRelST.target[of T1 T2 T_{Rel} S_{Rel}]}$

by simp

next

case (trans $P$, $Q$, $R$)

assume $(P, Q) \in (\text{indRelST} S_{Rel} T_{Rel})^*$ and $(Q, R) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

thus $(P, R) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

by simp

qed

next

fix $P, Q$

assume $(P, Q) \in (\text{indRelST} S_{Rel} T_{Rel})^*$

thus $P \sim [\cdot] < S_{Rel}, T_{Rel} > Q$

proof induct

from reflT show $P \sim [\cdot] < S_{Rel}, T_{Rel} > P$
using \texttt{indRelSTEQ-refl}\[1\] of \texttt{TRel SRel}
unfolding \texttt{refl-on-def}
by \texttt{simp}

next
case (step \(Q, R\))
assume \(P \sim \llbracket SRel, TRel \rrbracket Q\)
moreover assume \(Q \rel \llbracket SRel, TRel \rrbracket R\)
hence \(Q \sim \llbracket SRel, TRel \rrbracket R\)
by (induct, simp-all add: \texttt{indRelSTEQ.intros})
ultimately show \(P \sim \llbracket SRel, TRel \rrbracket R\)
by (rule \texttt{indRelSTEQ.trans})

qed

lemma (in encoding) \texttt{refl-symm-trans-closure-of-indRelST}:
\begin{align*}
\text{fixes } SRel & \:: ('procS \times 'procS) set \\
\text{and } TRel & \:: ('procT \times 'procT) set \\
\text{assumes } reflT & : \text{refl } TRel \\
\text{and } symmS & : \text{sym } SRel \\
\text{and } symmT & : \text{sym } TRel \\
\text{shows } \text{indRelSTEQ } SRel \ TRel = (\text{symcl } ((\text{indRelST } SRel \ TRel)^=))^+ \\
\text{proof --} \\
\text{have } (\text{symcl } ((\text{indRelST } SRel \ TRel)^=))^+ = (\text{symcl } (\text{indRelST } SRel \ TRel))^* \\
\text{by } (\text{rule refl-symm-trans-closure-is-symm-refl-trans-closure}\ [\text{where } \text{Rel} = \text{indRelST } SRel \ TRel])
\end{align*}
moreover from \texttt{symmS \text{symmT}} have \(\text{symcl } (\text{indRelST } SRel \ TRel) = \text{indRelST } SRel \ TRel\)
by \texttt{blast}
ultimately show \(\text{indRelSTEQ } SRel \ TRel = (\text{symcl } ((\text{indRelST } SRel \ TRel)^=))^+\)
by simp

qed

lemma (in encoding) \texttt{symm-closure-of-indRelRST}:
\begin{align*}
\text{fixes } SRel & \:: ('procS \times 'procS) set \\
\text{and } TRel & \:: ('procT \times 'procT) set \\
\text{assumes } reflT & : \text{refl } TRel \\
\text{and } symmS & : \text{sym } SRel \\
\text{and } symmT & : \text{sym } TRel \\
\text{shows } \text{indRelST } SRel \ TRel = \text{symcl } (\text{indRelRST } SRel \ TRel) \\
\text{and } \text{indRelSTEQ } SRel \ TRel = (\text{symcl } ((\text{indRelRST } SRel \ TRel)^=))^+ \\
\text{proof --} \\
\text{show } \text{indRelST } SRel \ TRel = \text{symcl } (\text{indRelRST } SRel \ TRel) \\
\text{proof auto} \\
\text{fix } P \ Q \\
\text{assume } P \rel \llbracket SRel, TRel \rrbracket Q \\
\text{thus } (P, Q) \in \text{symcl } (\text{indRelRST } SRel \ TRel) \\
\text{by (induct, simp-all add: symcl-def indRelRST.intros)}
\end{align*}
next
\begin{align*}
\text{fix } P \ Q \\
\text{assume } (P, Q) \in \text{symcl } (\text{indRelRST } SRel \ TRel) \\
\text{thus } P \rel \llbracket SRel, TRel \rrbracket Q \\
\text{proof (auto simp add: symcl-def indRelRST.simps)} \\
\text{fix } S \\
\text{show } \text{SourceTerm } S \rel \llbracket SRel, TRel \rrbracket \text{TargetTerm } (\llbracket S \rrbracket) \\
\text{by (rule indRelST.encR)}
\end{align*}
next
\begin{align*}
\text{fix } S1 \ S2 \\
\text{assume } (S1, S2) \in SRel \\
\text{thus } \text{SourceTerm } S1 \rel \llbracket SRel, TRel \rrbracket \text{SourceTerm } S2 \\
\text{by (rule indRelST.source)}
\end{align*}
next
  fix $T_1 \ T_2$
  assume $(T_1, T_2) \in \TRel$
  thus TargetTerm $T_1 \ R[\cdot]<SRel, \TRel> TargetTerm T_2$
     by (rule indRelST.target)
next
  fix $S$
  show TargetTerm $([S]) \ R[\cdot]<SRel, \TRel> SourceTerm S$
       by (rule indRelST.encL)
next
  fix $S_1 \ S_2$
  assume $(S_1, S_2) \in \SRel$
  with $\text{symmS}$ show SourceTerm $S_2 \ R[\cdot]<SRel, \TRel> SourceTerm S_1$
    unfolding $\text{sym-def}$
    by (simp add: indRelST.source)
next
  fix $T_1 \ T_2$
  assume $(T_1, T_2) \in \TRel$
  with $\text{symmT}$ show $(TargetTerm T_2, TargetTerm T_1) \in \indRelST \ SRel \ \TRel$
    unfolding $\text{sym-def}$
    by (simp add: indRelST.target)
qed

with $\text{reflT}$ show $\text{indRelSTEQ} \ SRel \ \TRel = (\text{symcl} ((\text{indRelRST} \ SRel \ \TRel)^=))^{+}$
  using $\text{refl-symm-trans-closure-is-symm-refl-trans-closure}$[where $\text{Rel} = \text{indRelRST} \ SRel \ \TRel]$
  refl-trans-closure-of-indRelST
  by simp
qed

lemma (in encoding) $\text{symm-closure-of-indRelLST}$:
  fixes $\text{SRel} :: ('\text{procS} \times '\text{procS}) \ \text{set}$
  and $\text{TRel} :: ('\text{procT} \times '\text{procT}) \ \text{set}$
  assumes $\text{reflT}$: refl $\text{TRel}$
    and $\text{symmS}$: sym $\text{SRel}$
    and $\text{symmT}$: sym $\text{TRel}$
  shows $\text{indRelLST} \ SRel \ \TRel = \text{symcl} \ (\text{indRelLST} \ SRel \ \TRel)$
    and $\text{indRelSTEQ} \ SRel \ \TRel = (\text{symcl} ((\text{indRelLST} \ SRel \ \TRel)^=))^{+}$
proof --
  show $\text{indRelLST} \ SRel \ \TRel = \text{symcl} \ (\text{indRelLST} \ SRel \ \TRel)$
    proof
    auto
    fix $P \ Q$
    assume $P \ R[\cdot]<SRel, \TRel> Q$
    thus $(P, Q) \in \text{symcl} \ (\text{indRelLST} \ SRel \ \TRel)$
       by (induct, simp-all add: $\text{symcl-def}$ indRelLST.intros)
next
  fix $P \ Q$
  assume $(P, Q) \in \text{symcl} \ (\text{indRelLST} \ SRel \ \TRel)$
  thus $P \ R[\cdot]<SRel, \TRel> Q$
    proof (auto simp add: $\text{symcl-def}$ indRelLST.simps)
    fix $S$
    show $\text{SourceTerm} S \ R[\cdot]<SRel, \TRel> TargetTerm ([S])$
       by (rule indRelST.encR)
next
  fix $S_1 \ S_2$
  assume $(S_1, S_2) \in \SRel$
  thus $\text{SourceTerm} S_1 \ R[\cdot]<SRel, \TRel> \text{SourceTerm} S_2$
    by (rule indRelST.source)
next
  fix $T_1 \ T_2$
  assume $(T_1, T_2) \in \TRel$
  thus $\text{TargetTerm} T_1 \ R[\cdot]<SRel, \TRel> \text{TargetTerm} T_2$

by (rule indRelST.target)

next
fix S
show TargetTerm ([S]) \( R[[S],T]\) SourceTerm S
  by (rule indRelST.encL)

next
fix S1 S2
assume (S1, S2) \(\in\) SRel
with symmS show SourceTerm S2 \( R[[S],T]\) SourceTerm S1
  unfolding sym-def
  by (simp add: indRelST.source)

next
fix T1 T2
assume (T1, T2) \(\in\) TRel
with symmT show TargetTerm T2 \( R[[S],T]\) TargetTerm T1
  unfolding sym-def
  by (simp add: indRelST.target)

qed

lemma (in encoding) symm-trans-closure-of-indRelRSTPO:
  fixes SRel :: \('procS \times 'procS\) set
  and TRel :: \('procT \times 'procT\) set
  assumes symmS:: sym SRel
  and symmT:: sym TRel
  shows indRelSTEQ SRel TRel = \((\text{symcl ((indRelLST SRel TRel)=\())\}\)]
proof auto
  fix P Q
  assume P \(\sim\) Q
  thus \((P, Q) \in (\text{symcl ((indRelRSTPO SRel TRel)=\())\}\)]
proof induct
  case (encR S)
  show \((\text{SourceTerm S, TargetTerm ([S]]) \in (\text{symcl ((indRelRSTPO SRel TRel)=\())\}\)]
    using indRelRSTPO.encR[of S SRel TRel]
    unfolding symcl-def
    by auto

next
  case (encL S)
  show \((\text{TargetTerm ([S]), SourceTerm S}) \in (\text{symcl ((indRelRSTPO SRel TRel)=\())\}\)]
    using indRelRSTPO.encR[of S SRel TRel]
    unfolding symcl-def
    by auto

next
  case (source S1 S2)
  assume (S1, S2) \(\in\) SRel
  thus \((\text{SourceTerm S1, SourceTerm S2}) \in (\text{symcl ((indRelRSTPO SRel TRel)=\())\}\)]
    using indRelRSTPO.source[of S1 S2 SRel TRel]
    unfolding symcl-def
    by auto

next
  case (target T1 T2)
  assume (T1, T2) \(\in\) TRel
  thus \((\text{TargetTerm T1, TargetTerm T2}) \in (\text{symcl ((indRelRSTPO SRel TRel)=\())\}\)]
    using indRelRSTPO.target[of T1 T2 TRel SRel]
    unfolding symcl-def
    by auto

qed
next
  case (trans P Q R)
  assume (P, Q) ∈ (symcl (indRelRSTPO SRel TRel))↑
    and (Q, R) ∈ (symcl (indRelRSTPO SRel TRel))↑
    thus (P, R) ∈ (symcl (indRelRSTPO SRel TRel))↑
  by simp
qed

next
fix P Q
assume (P, Q) ∈ (symcl (indRelRSTPO SRel TRel))↑
thus P ∼[\cdot]<SRel,TRel> Q
proof induct
fix Q
assume (P, Q) ∈ symcl (indRelRSTPO SRel TRel)
thus P ∼[\cdot]<SRel,TRel> Q
proof (cases P ≲[\cdot]<R<\cdot, SRel, TRel> Q, simp-all add: symcl-def)
  assume P ≲[\cdot]<R<\cdot, SRel, TRel> Q
  thus P ∼[\cdot]<SRel, TRel> Q
  proof
    show SourceTerm S ∼[\cdot]<SRel, TRel> TargetTerm ([S])
      by (rule indRelSTEQ.encR)
  next
    case (source S1 S2)
    assume (S1, S2) ∈ SRel
    thus SourceTerm S1 ∼[\cdot]<SRel, TRel> SourceTerm S2
      by (rule indRelSTEQ.source)
  next
    case (target T1 T2)
    assume (T1, T2) ∈ TRel
    thus TargetTerm T1 ∼[\cdot]<SRel, TRel> TargetTerm T2
      by (rule indRelSTEQ.target)
  next
    case (trans P Q R)
    assume R ∼[\cdot]<\cdot<\cdot, SRel, TRel> Q
    thus R ∼[\cdot]<SRel, TRel> P
    proof
      case (encR S)
      show TargetTerm ([S]) ∼[\cdot]<SRel, TRel> SourceTerm S
        by (rule indRelSTEQ.encL)
      next
      case (source S1 S2)
      assume (S1, S2) ∈ SRel
      with symmS show SourceTerm S2 ∼[\cdot]<SRel, TRel> SourceTerm S1
        unfolding symm-def
        by (simp add: indRelSTEQ.source)
      next
      case (target T1 T2)
      assume (T1, T2) ∈ TRel
      with symmT show TargetTerm T2 ∼[\cdot]<SRel, TRel> TargetTerm T1
        unfolding symm-def
        by (simp add: indRelSTEQ.target)
      next
      case (trans P Q R)
      assume R ∼[\cdot]<\cdot<\cdot, SRel, TRel> Q
      thus R ∼[\cdot]<SRel, TRel> P

by (rule indRelSTEQ.trans)
qed
qed
next

\begin{case}
\text{\textbf{step}} Q R
assume \( P \sim_{[\cdot]} SRel, TRel \) \( Q \)
moreover assume \((Q, R) \in \text{symcl} (\text{indRelRSTPO} SRel TRel)\)
hence \( Q \sim_{[\cdot]} SRel, TRel \)
\end{case}

\begin{proof}
(auto simp add: symcl-def)
assume \( Q \leq_{[\cdot]} SRel, TRel \)
thus \( Q \sim_{[\cdot]} SRel, TRel \)
\end{proof}

\begin{proof}
(induct, simp add: indRelSTEQ.encR, simp add: indRelSTEQ.source, simp add: indRelSTEQ.target)
\end{proof}

\begin{case}
assume \( P \sim_{[\cdot]} SRel, TRel \) \( Q \) and \( Q \sim_{[\cdot]} SRel, TRel \)
\end{case}

\begin{proof}
by (rule indRelSTEQ.trans)
\end{proof}

\begin{next}
\end{next}

\begin{next}
assume \( R \leq_{[\cdot]} SRel, TRel \)
\end{next}

\begin{proof}
(induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ.source, simp add: indRelSTEQ.target)
\end{proof}

\begin{case}
assume \( P \sim_{[\cdot]} SRel, TRel \) \( Q \) and \( Q \sim_{[\cdot]} SRel, TRel \)
\end{case}

\begin{proof}
by (rule indRelSTEQ.trans)
\end{proof}

\begin{next}
\end{next}

\begin{next}
\end{next}

\begin{lemma}
\begin{proof}
\end{proof}
\end{lemma}
case (source S1 S2)
assume (S1, S2) ∈ SRel
thus (SourceTerm S1, SourceTerm S2) ∈ (symcl (indRelLSTPO SRel TRel))
using indRelLSTPO.of SRel TRel
unfolding symcl-def
by blast

next

case (target T1 T2)
assume (T1, T2) ∈ TRel
thus (TargetTerm T1, TargetTerm T2) ∈ (symcl (indRelLSTPO SRel TRel))
using indRelLSTPO.of T1 T2 SRel TRel
unfolding symcl-def
by blast

next

case (trans P Q R)
assume P ~< SRel TRel > Q
and Q ~< SRel TRel > R
thus P ~< SRel TRel > R
by (rule indRelSTEQ.trans)
qed

next

fix P Q

assume (P, Q) ∈ (symcl (indRelLSTPO SRel TRel))
thus P ~< SRel TRel > Q

proof induct

fix Q

assume (P, Q) ∈ symcl (indRelLSTPO SRel TRel)
thus P ~< SRel TRel > Q
unfolding symcl-def

proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ.source,
          simp add: indRelSTEQ.target)

  case (trans P Q R)
  assume P ~< SRel TRel > Q and Q ~< SRel TRel > R
  thus P ~< SRel TRel > R
  by (rule indRelSTEQ.trans)
qed

next

assume Q ~< SRel TRel > P
hence Q ~< SRel TRel > P

proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ.source,
          simp add: indRelSTEQ.target)

  case (trans P Q R)
  assume P ~< SRel TRel > Q and Q ~< SRel TRel > R
  thus P ~< SRel TRel > R
  by (rule indRelSTEQ.trans)
qed

next

with symmS symmT show P ~< SRel TRel > Q
using indRelSTEQ-symm.of SRel TRel
unfolding sym-def
by blast

next

case (step Q R)
assume P ~< SRel TRel > Q

moreover assume (Q, R) ∈ symcl (indRelLSTPO SRel TRel)

hence Q ~< SRel TRel > R
unfolding symcl-def

proof auto
assume $Q \lesssim L < SRel, TRel > R$
thus $Q \sim L < SRel, TRel > R$
proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ. source, simp add: indRelSTEQ. target)
case (trans $P Q R$
assume $P \sim L < SRel, TRel > Q$ and $Q \sim L < SRel, TRel > R$
thus $P \sim L < SRel, TRel > R$
by (rule indRelSTEQ. trans)
qed

next
assume $R \lesssim L < SRel, TRel > Q$

hence $R \sim L < SRel, TRel > Q$
proof (induct, simp add: indRelSTEQ.encL, simp add: indRelSTEQ. source, simp add: indRelSTEQ. target)
case (trans $P Q R$
assume $P \sim L < SRel, TRel > Q$ and $Q \sim L < SRel, TRel > R$
thus $P \sim L < SRel, TRel > R$
by (rule indRelSTEQ. trans)
qed

with symmS symmT show $Q \sim L < SRel, TRel > R$
using indRelSTEQ-symm[of $SRel TRel$]
unfolding sym-def
by blast
qed

ultimately show $P \sim L < SRel, TRel > R$
by (rule indRelSTEQ. trans)

qed

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

lemma (in encoding) indRelRST-to-SRel:
fixes $SRel :: (\'procS \times \'procS) \ set$
and $TRel :: (\'procT \times \'procT) \ set$
and $SP SQ :: \'procS$
assumes rel: SourceTerm $SP \ R< SRel, TRel > SourceTerm SQ$
shows $(SP, SQ) \in SRel$
using rel
by (simp add: indRelRST. simps)

lemma (in encoding) indRelRST-to-TRel:
fixes $SRel :: (\'procS \times \'procS) \ set$
and $TRel :: (\'procT \times \'procT) \ set$
and $TP TQ :: \'procT$
assumes rel: TargetTerm $TP \ R< SRel, TRel > TargetTerm TQ$
shows $(TP, TQ) \in TRel$
using rel
by (simp add: indRelRST. simps)

lemma (in encoding) indRelLST-to-SRel:
fixes $SRel :: (\'procS \times \'procS) \ set$
and $TRel :: (\'procT \times \'procT) \ set$
and $SP SQ :: \'procS$
assumes rel: SourceTerm $SP \ L< SRel, TRel > SourceTerm SQ$
shows $(SP, SQ) \in SRel$
using rel
by (simp add: indRelLST. simps)

lemma (in encoding) indRelLST-to-TRel:
proof

lemma (in encoding) indRelST-to-SRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and SP SQ :: 'procS
  assumes rel: SourceTerm SP R[[<SRel,TRel>]] SourceTerm SQ
  shows (SP, SQ) ∈ SRel
  using rel
  by (simp add: indRelLST.simps)

lemma (in encoding) indRelST-to-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and P Q :: 'procT
  assumes rel: TargetTerm TP R[[<SRel,TRel>]] TargetTerm TQ
  shows (TP, TQ) ∈ TRel
  using rel
  by (simp add: indRelLST.simps)

lemma (in encoding) indRelRSTPO-to-SRel-and-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and P Q :: 'procT
  assumes rel: TargetTerm TP R[[<SRel,TRel>]] TargetTerm TQ
  shows (TP, TQ) ∈ TRel
  using rel
  by (simp add: indRelLST.simps)

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

proof induct
  case (encR S)
  show ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → (SP, SQ) ∈ SRel⁺
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → False
  and ∀ TP TQ. TP ∈ T SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → (TP, TQ) ∈ TRel⁺
    by simp⁺
  have ([S], [S]) ∈ TRel⁺
    by simp
  moreover have ([S], [S]) ∈ TRel⁺
    by simp
  ultimately show ∀ SP TQ. SP ∈ S SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → (SP, SQ) ∈ SRel⁺
    (S P, S) ∈ SRel⁺ ∧ ([S], TQ) ∈ TRel⁺
    by blast

next
  case (source S1 S2)
  assume (S1, S2) ∈ SRel
  thus ∀ SP SQ. SP ∈ S SourceTerm S1 ∧ SQ ∈ S SourceTerm S2 → (SP, SQ) ∈ SRel⁺
    by simp
  show ∀ SP TQ. SP ∈ S SourceTerm S1 ∧ TQ ∈ T SourceTerm S2 →
\[(\exists S. (SP, S) \in SRel^* \land ([S], TQ) \in TRel^*)\]

and \(\forall TP SQ. TP \in T SourceTerm S1 \land SQ \in S SourceTerm S2 \longrightarrow False\)

and \(\forall TP TQ. TP \in T SourceTerm S1 \land TQ \in T SourceTerm S2 \longrightarrow (TP, TQ) \in TRel^+\)

by simp

next

case (target T1 T2)

\[\begin{array}{l}
\text{show } \forall SP SQ. SP \in STargetTerm T1 \land SQ \in STargetTerm T2 \longrightarrow (SP, SQ) \in SRel^+ \\
\text{and } \forall SP TQ. SP \in STargetTerm T1 \land TQ \in TTargetTerm T2 \longrightarrow (\exists S. (SP, S) \in SRel^* \land ([S], TQ) \in TRel^*) \\
\text{and } \forall TP SQ. TP \in T TargetTerm T1 \land SQ \in S TargetTerm T2 \longrightarrow False \\
\text{by simp+}
\end{array}\]

assum (T1, T2) \in TRel

thus \(\forall TP TQ. TP \in T TargetTerm T1 \land TQ \in T TargetTerm T2 \longrightarrow (TP, TQ) \in TRel^+\)

by simp

next

case (trans P Q R)

\[\begin{array}{l}
\text{assume } A1: \forall SP SQ. SP \in S P \land SQ \in S Q \longrightarrow (SP, SQ) \in SRel^+ \\
\text{and } A2: \forall SP TQ. SP \in S P \land TQ \in T Q \longrightarrow (\exists S. (SP, S) \in SRel^* \land ([S], TQ) \in TRel^*) \\
\text{and } A3: \forall TP SQ. TP \in T P \land SQ \in S Q \longrightarrow False \\
\text{and } A4: \forall TP TQ. TP \in T P \land TQ \in T Q \longrightarrow (TP, TQ) \in TRel^+ \\
\text{and } A5: \forall SQ SR. SQ \in S Q \land SR \in S R \longrightarrow (SQ, SR) \in SRel^+ \\
\text{and } A6: \forall SQ TR. SQ \in S Q \land TR \in T R \longrightarrow (\exists S. (SQ, S) \in SRel^* \land ([S], TR) \in TRel^*) \\
\text{and } A7: \forall TQ SR. TQ \in T Q \land SR \in S R \longrightarrow False \\
\text{and } A8: \forall TQ TR. TQ \in T Q \land TR \in T R \longrightarrow (TQ, TR) \in TRel^+ \\
\text{show } \forall SP SR. SP \in S P \land SR \in S R \longrightarrow (SP, SR) \in SRel^+
\end{array}\]

\[\begin{array}{l}
\text{proof } \text{clarify} \\
\text{fix } SP SR \\
\text{assume } A9: SP \in S P \text{ and } A10: SR \in S R \\
\text{show } (SP, SR) \in SRel^+ \\
\text{proof } (\text{cases } Q) \\
\text{case } (SourceTerm SQ) \\
\text{assume } A11: SQ \in S Q \\
\text{with } A1 A9 \text{ have } (SP, SQ) \in SRel^+ \\
\text{by simp}
\end{array}\]

moreover from A5 A10 A11 have \((SQ, SR) \in SRel^+\)

by simp

ultimately show \((SP, SR) \in SRel^+\)

by simp

next

case (TargetTerm TQ)

\[\begin{array}{l}
\text{assume } TQ \in T Q \\
\text{with } A7 A10 \text{ show } (SP, SR) \in SRel^+ \\
\text{by blast}
\end{array}\]

qed

qed

show \(\forall SP TR. SP \in S P \land TR \in T R \longrightarrow (\exists S. (SP, S) \in SRel^* \land ([S], TR) \in TRel^*)\)

\[\begin{array}{l}
\text{proof } \text{clarify} \\
\text{fix } SP TR \\
\text{assume } A9: SP \in S P \text{ and } A10: TR \in T R \\
\text{show } \exists S. (SP, S) \in SRel^* \land ([S], TR) \in TRel^* \\
\text{proof } (\text{cases } Q) \\
\text{case } (SourceTerm SQ) \\
\text{assume } A11: SQ \in S Q \\
\text{with } A6 A10 \text{ obtain } S \text{ where } A12: (SQ, S) \in SRel^* \\
\text{and } A13: ([S], TR) \in TRel^* \\
\text{by blast}
\end{array}\]

from A1 A9 A11 have \((SP, SQ) \in SRel^+\)

by simp

from this A12 have \((SP, S) \in SRel^*\)

by simp
\[ A13 \text{ show } \exists S. (SP, S) \in SRel^* \land ([S], TR) \in TRel^* \]

by blast

next

case (TargetTerm TQ)

assume A11: TQ \in T Q

with A2 A9 obtain S where A12: (SP, S) \in SRel^*

and A13: ([S], TQ) \in TRel^*

by blast

from A8 A10 A11 have (TQ, TR) \in TRel^*

by simp

with A13 have ([S], TR) \in TRel^*

by simp

with A12 show \exists S. (SP, S) \in SRel^* \land ([S], TR) \in TRel^*

by blast

qed

show \forall TP SR. TP \in T P \land SR \in S R \rightarrow False

proof clarify

fix TP SR

assume A9: TP \in T P and A10: SR \in S R

show False

proof (cases Q)

case (SourceTerm SQ)

assume SQ \in S Q

with A3 A9 show False

by blast

next

case (TargetTerm TQ)

assume TQ \in T Q

with A7 A10 show False

by blast

qed

show \forall TP TR. TP \in T P \land TR \in T R \rightarrow (TP, TR) \in TRel^+

proof clarify

fix TP TR

assume A9: TP \in T P and A10: TR \in T R

show (TP, TR) \in TRel^+

proof (cases Q)

case (SourceTerm SQ)

assume SQ \in S Q

with A3 A9 show (TP, TR) \in TRel^+

by blast

next

case (TargetTerm TQ)

assume A11: TQ \in T Q

with A4 A9 have (TP, TQ) \in TRel^+

by simp

moreover from A8 A10 A11 have (TQ, TR) \in TRel^+

by simp

ultimately show (TP, TR) \in TRel^+

by simp

qed

qed

lemma (in encoding) indRelSTPO-to-SRel-and-TRel:

fixes SRel :: (′procS × ′procS) set

and TRel :: (′procT × ′procT) set

and P Q :: (′procS, ′procT) Proc

assumes P \langle SRel, TRel > Q
shows \( \forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow (SP, SQ) \in SRel^+ \)
and \( \forall SP TQ. SP \in S P \land TQ \in T Q \rightarrow \text{False} \)
and \( \forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow (\exists S. (TP, [S]) \in TRel^* \land (S, SQ) \in SRel^*) \)
and \( \forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in TRel^+ \)
using assms

proof induct

\text{case (encL S)}

\text{show } \forall SP SQ. SP \in S TargetTerm ([S]) \land SQ \in S SourceTerm S \rightarrow (SP, SQ) \in SRel^+
and \( \forall SP TQ. SP \in S TargetTerm ([S]) \land TQ \in T SourceTerm S \rightarrow \text{False} \)
and \( \forall TP SQ. TP \in T TargetTerm ([S]) \land SQ \in S SourceTerm S \rightarrow (TP, TQ) \in TRel^+ \)
by simp+

\text{have } ([S], [S]) \in TRel^*
by simp

moreover have \((S, S) \in SRel^*

by simp

ultimately show \( \forall TP SQ. TP \in T TargetTerm ([S]) \land SQ \in S SourceTerm S \rightarrow (TP, SQ) \in TRel^* \land (S, SQ) \in SRel^*)
by blast

next

\text{case (source S1 S2)}
assume \((S1, S2) \in SRel\)
thus \( \forall SP SQ. SP \in S SourceTerm S1 \land SQ \in S SourceTerm S2 \rightarrow (SP, SQ) \in SRel^+ \)
by simp

\text{show } \forall SP TQ. SP \in S SourceTerm S1 \land TQ \in T SourceTerm S2 \rightarrow \text{False} \)
and \( \forall TP SQ. TP \in T SourceTerm S1 \land SQ \in S SourceTerm S2 \rightarrow (\exists S. (TP, [S]) \in TRel^* \land (S, SQ) \in SRel^*) \)
and \( \forall TP TQ. TP \in T SourceTerm S1 \land TQ \in T SourceTerm S2 \rightarrow (TP, TQ) \in TRel^+ \)
by simp+

next

\text{case (target T1 T2)}

\text{show } \forall SP SQ. SP \in S TargetTerm T1 \land SQ \in S TargetTerm T2 \rightarrow (SP, SQ) \in SRel^+
and \( \forall SP TQ. SP \in S TargetTerm T1 \land TQ \in T TargetTerm T2 \rightarrow \text{False} \)
and \( \forall TP SQ. TP \in T TargetTerm T1 \land SQ \in S TargetTerm T2 \rightarrow (\exists S. (TP, [S]) \in TRel^* \land (S, SQ) \in SRel^*) \)
by simp+

assume \((T1, T2) \in TRel\)
thus \( \forall TP TQ. TP \in T TargetTerm T1 \land TQ \in T TargetTerm T2 \rightarrow (TP, TQ) \in TRel^+ \)
by simp

next

\text{case (trans P Q R)}
assume \(A1: \forall SP SQ. SP \in S P \land SQ \in S Q \rightarrow (SP, SQ) \in SRel^+ \)
and \(A2: \forall SP TQ. SP \in S P \land TQ \in T Q \rightarrow \text{False} \)
and \(A3: \forall TP SQ. TP \in T P \land SQ \in S Q \rightarrow (TP, SQ) \in SRel^+ \)
and \(A4: \forall TP TQ. TP \in T P \land TQ \in T Q \rightarrow (TP, TQ) \in TRel^+ \)
and \(A5: \forall SQ SR. SQ \in S Q \land SR \in S R \rightarrow (SQ, SR) \in SRel^+ \)
and \(A6: \forall SQ TR. SQ \in S Q \land TR \in T R \rightarrow \text{False} \)
and \(A7: \forall TQ SR. TQ \in T Q \land SR \in S R \rightarrow (TQ, SR) \in TRel^+ \)
and \(A8: \forall TQ TR. TQ \in T Q \land TR \in T R \rightarrow (TQ, TR) \in TRel^+ \)
show \( \forall SP SR. SP \in S P \land SR \in S R \rightarrow (SP, SR) \in SRel^+ \)

proof clarify

fix SP SR
assume \(A9: SP \in S P \) and \(A10: SR \in S R \)
show \((SP, SR) \in SRel^+ \)

proof (cases Q)

\text{case (SourceTerm SQ)}
assume \(A11: SQ \in S Q \)
with \(A1 \quad A9\) have \((SP, SQ) \in SRel^+ \)
by simp

moreover from \(A5 \quad A10 \quad A11\) have \((SQ, SR) \in SRel^+ \)
by simp
ultimately show \((SP, SR) \in SRel^+\)
   by \texttt{simp}

next
case \((\text{TargetTerm} TQ)\)
assume \(TQ \in T Q\)
with \(A2\) \(A9\) show \((SP, SR) \in SRel^+\)
   by \texttt{blast}

qed

qed

show \(\forall SP TR. SP \in S P \land TR \in T R \implies False\)
proof \texttt{clarify}
fix \(SP TR\)
assume \(A9: SP \in S P\) and \(A10: TR \in T R\)
show \(False\)
proof
   \((\text{cases } Q)\)
   case \((\text{SourceTerm SQ})\)
   assume \(SQ \in S Q\)
   with \(A6\) \(A10\) show \(False\)
      by \texttt{blast}

next
case \((\text{TargetTerm TQ})\)
assume \(TQ \in T Q\)
with \(A2\) \(A9\) show \(False\)
   by \texttt{blast}

qed

qed

show \(\forall TP SR. TP \in T P \land SR \in S R \implies (\exists S. (TP, [S]) \in TRel^* \land (S, SR) \in SRel^*)\)
proof \texttt{clarify}
fix \(TP SR\)
assume \(A9: TP \in T P\) and \(A10: TR \in T R\)
show \(\exists S. (TP, [S]) \in TRel^* \land (S, SR) \in SRel^*\)
proof
   \((\text{cases } Q)\)
   case \((\text{SourceTerm SQ})\)
   assume \(A11: SQ \in S Q\)
   with \(A3\) \(A9\) obtain \(S\) where \(A12: (TP, [S]) \in TRel^*\) and \(A13: (S, SQ) \in SRel^*\)
      by \texttt{blast}
   from \(A5\) \(A10\) \(A11\) have \((SQ, SR) \in SRel^*\)
      by \texttt{simp}
   with \(A13\) have \((S, SR) \in SRel^*\)
      by \texttt{simp}
   with \(A12\) show \(\exists S. (TP, [S]) \in TRel^* \land (S, SR) \in SRel^*\)
      by \texttt{blast}

next
case \((\text{TargetTerm TQ})\)
assume \(A11: TQ \in T Q\)
with \(A7\) \(A10\) obtain \(S\) where \(A12: (TQ, [S]) \in TRel^*\) and \(A13: (S, SQ) \in SRel^*\)
      by \texttt{blast}
   from \(A4\) \(A9\) \(A11\) have \((TP, TQ) \in TRel^*\)
      by \texttt{simp}
   from this \(A12\) have \((TP, [S]) \in TRel^*\)
      by \texttt{simp}
   with \(A13\) show \(\exists S. (TP, [S]) \in TRel^* \land (S, SR) \in SRel^*\)
      by \texttt{blast}

qed

qed

show \(\forall TP TR. TP \in T P \land TR \in T R \implies (TP, TR) \in TRel^+\)
proof \texttt{clarify}
fix \(TP TR\)
assume \(A9: TP \in T P\) and \(A10: TR \in T R\)
show \((TP, TR) \in TRel^+\)
proof \((\text{cases } Q)\)
If \( \text{indRelRSTPO}, \text{indRelLSTPO}, \) or \( \text{indRelSTPO} \) 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 \( SRel :: ('procS \times 'procS) \) set
and \( Rel :: ('procS, 'procT) \) set
assumes preservation: rel-preserves-barbs \( \text{Rel} (\text{STCalWB SWB TWB}) \)
and sourceInRel: \( \forall S1 S2. (S1, S2) \in SRel \rightarrow (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel} \)
sows rel-preserves-barbs \( SRel \) SWB

**Proof**
clarify
fix \( SP SQ a \)
assume \( (SP, SQ) \in SRel \)
with sourceInRel have \( (\text{SourceTerm} SP, \text{SourceTerm} SQ) \in \text{Rel} \)
by blast
moreover assume \( SP \downarrow<\text{SWB}>a \)
hence \( \text{SourceTerm} SP \downarrow.a \)
by simp
ultimately have \( \text{SourceTerm} SQ \downarrow.a \)
by simp
thus \( SQ \downarrow<\text{SWB}>a \)
by simp
qed

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

fixes \( SRel :: ('procS \times 'procS) \) set
and \( TRel :: ('procT \times 'procT) \) set
assumes preservation: rel-preserves-barbs \( \text{(indRelRSTPO SRel TRel) (STCalWB SWB TWB)} \)
sows rel-preserves-barbs \( SRel \) SWB
and rel-preserves-barbs \( TRel \) TWB

**Proof**
–
show rel-preserves-barbs \( SRel \) SWB
using preservation rel-with-source-impl-SRel-preserves-barbs[where \( \text{Rel} = \text{SRel} \) and \( \text{SRel} = \text{SRel} \)]
by (simp add: indRelRSTPO.source)
next
show rel-preserves-barbs \( TRel \) TWB
using preservation rel-with-target-impl-TRel-preserves-barbs[where \( \text{Rel} = \text{SRel} \) and \( \text{TRel} = \text{TRel} \)]
by (simp add: indRelRSTPO.target)
qed

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

fixes \( SRel :: ('procS \times 'procS) \) set
and $T_{rel}$ :: $(\texttt{procT} \times \texttt{procT})$ set
assumes preservation: rel-preserves-bars $\cdot (\texttt{indRelLSTPO } S_{rel} T_{rel})$ $(STCalWB SWB TWB)$
shows rel-preserves-bars $S_{rel}$ SWB
and rel-preserves-bars $T_{rel}$ TWB
proof –
show rel-preserves-bars $S_{rel}$ SWB
  using preservation rel-with-source-impl-SRel-preserves-barbs[where
  $Rel=\texttt{indRelLSTPO } S_{rel} T_{rel}$ and $S_{rel}=S_{rel}$]
  by (simp add: indRelLSTPO.source)
next
show rel-preserves-bars $T_{rel}$ TWB
  using preservation rel-with-target-impl-TRel-preserves-barbs[where
  $Rel=\texttt{indRelLSTPO } S_{rel} T_{rel}$ and $T_{rel}=T_{rel}$]
  by (simp add: indRelLSTPO.target)
qed

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-preserve-barbs:
  fixes $S_{rel}$ :: $(\texttt{procS} \times \texttt{procS})$ set
  and $T_{rel}$ :: $(\texttt{procT} \times \texttt{procT})$ set
  assumes preservation: rel-preserves-bars $(\texttt{indRelSTEQ } S_{rel} T_{rel})$ $(STCalWB SWB TWB)$
  shows rel-preserves-bars $S_{rel}$ SWB
  and rel-preserves-bars $T_{rel}$ TWB
proof
clarify
fix $SP$ $SQ$ a $SP'$
assume $(SP, SQ) \in S_{rel}$
with sourceInRel have (SourceTerm $SP$, SourceTerm $SQ$) $\in$ $Rel$
  by blast
moreover assume $SP \rightarrow (Calculus SWB)* SP'$ and $SP'\downarrow<SWB>a$
hence SourceTerm $SP\downarrow.a$
  by blast
ultimately have SourceTerm $SQ\downarrow.a$
  using preservation weak-preservation-of-barbs-in-barbed-encoding[where $Rel=Rel$]
  by blast
thus $SQ\downarrow<SWB>a$
  by simp
qed

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-weakly-preserves-barbs:
  fixes $S_{rel}$ :: $(\texttt{procS} \times \texttt{procS})$ set
  and $Rel$ :: $(\texttt{procS}, \texttt{procT})$ Proc $\times$ $(\texttt{procS}, \texttt{procT})$ Proc set
  assumes preservation: rel-weakly-preserves-barbs $Rel$ $(STCalWB SWB TWB)$
  and sourceInRel: $\forall S_1 S_2. (S_1, S_2) \in S_{rel} \rightarrow (SourceTerm S_1, SourceTerm S_2) \in Rel$
  shows rel-weakly-preserves-barbs $S_{rel}$ SWB
proof clarify
fix $SP$ $SQ$ a $SP'$
assume $(SP, SQ) \in S_{rel}$
with sourceInRel have (SourceTerm $SP$, SourceTerm $SQ$) $\in$ $Rel$
  by blast
moreover assume $SP \rightarrow (Calculus SWB)* SP'$ and $SP'\downarrow<SWB>a$
  hence $SP\downarrow.a$
  by blast
ultimately have $SQ\downarrow.a$
  using preservation weak-preservation-of-barbs-in-barbed-encoding[where $Rel=Rel$]
  by blast
thus $SQ\downarrow<SWB>a$
  by simp
qed

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes $S_{rel}$ :: $(\texttt{procS} \times \texttt{procS})$ set
  and $T_{rel}$ :: $(\texttt{procT} \times \texttt{procT})$ set
  assumes preservation: rel-weakly-preserves-barbs $(\texttt{indRelRSTPO } S_{rel} T_{rel})$ $(STCalWB SWB TWB)$
  shows rel-weakly-preserves-barbs $S_{rel}$ SWB
  and rel-weakly-preserves-barbs $T_{rel}$ TWB
proof –

show rel-weakly-preserves-barbs SRel SWB
  using preservation rel-with-source-impl-SRel-weakly-preserves-barbs[where
    \(\text{Rel} = \text{indRelRSTPO SRel TRel \ and \ SRel = SRel}\)]
  by (simp add: indRelRSTPO.source)

next

show rel-weakly-preserves-barbs TRel TWB
  using preservation rel-with-target-impl-TRel-weakly-preserves-barbs[where
    \(\text{Rel} = \text{indRelRSTPO SRel TRel \ and \ TRel = TRel}\)]
  by (simp add: indRelRSTPO.target)

qed

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: \("\text{procS} \times \text{procS}\) set
  and TRel :: \("\text{procS} \times \text{procT}\) set
    \(\text{assumes preservation: rel-weakly-preserves-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)}\)
    \(\text{shows rel-weakly-preserves-barbs SRel SWB \ and rel-weakly-preserves-barbs TRel TWB}\)
    \(\text{proof –}\)
    \(\text{show rel-weakly-preserves-barbs SRel SWB}\)
      using preservation rel-with-source-impl-SRel-weakly-preserves-barbs[where
        \(\text{Rel} = \text{indRelLSTPO SRel TRel \ and \ SRel = SRel}\)]
      by (simp add: indRelLSTPO.source)
    next
    \(\text{show rel-weakly-preserves-barbs TRel TWB}\)
      using preservation rel-with-target-impl-TRel-weakly-preserves-barbs[where
        \(\text{Rel} = \text{indRelLSTPO SRel TRel \ and \ TRel = TRel}\)]
      by (simp add: indRelLSTPO.target)
    qed

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs:
  fixes SRel :: \("\text{procS} \times \text{procS}\) set
  and TRel :: \("\text{procS} \times \text{procT}\) set
    \(\text{assumes reflection: rel-weakly-preserves-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)}\)
    \(\text{shows rel-weakly-preserves-barbs SRel SWB \ and rel-weakly-preserves-barbs TRel TWB}\)
    \(\text{proof –}\)
    \(\text{show rel-weakly-preserves-barbs SRel SWB}\)
      using preservation rel-with-source-impl-SRel-weakly-preserves-barbs[where
        \(\text{Rel} = \text{indRelSTEQ SRel TRel \ and \ SRel = SRel}\)]
      by (simp add: indRelSTEQ.source)
    next
    \(\text{show rel-weakly-preserves-barbs TRel TWB}\)
      using preservation rel-with-target-impl-TRel-weakly-preserves-barbs[where
        \(\text{Rel} = \text{indRelSTEQ SRel TRel \ and \ TRel = TRel}\)]
      by (simp add: indRelSTEQ.target)
    qed

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

lemma (in encoding-wrt-barbs) rel-with-source-impl-SRel-reflects-barbs:
  fixes SRel :: \("\text{procS} \times \text{procS}\) set
  and Rel :: \("\text{procS} \times \text{procT}\) Proc \times \("\text{procS} \times \text{procT}\) Proc\) set
    \(\text{assumes reflection: rel-reflects-barbs Rel (STCalWB SWB TWB)}\)
    \(\text{and sourceInRel: \(\forall S1 S2. (S1, S2) \in SRel \rightarrow (SourceTerm S1, SourceTerm S2) \in \text{Rel}\)}\)
    \(\text{shows rel-reflects-barbs SRel SWB}\)
    \(\text{proof clarify}\)
    \(\text{fix SP SQ a}\)
    \(\text{assume (SP, SQ) \in SRel}\)
    \(\text{with sourceInRel have (SourceTerm SP, SourceTerm SQ) \in \text{Rel}}\)
by blast
moreover assume \( SQ\downarrow<\text{SWB}>a \)

hence SourceTerm \( SQ\downarrow\ a \)

by simp

ultimately have SourceTerm \( SP\downarrow\ a \)

using reflection reflection-of-bars-in-barbed-encoding[where \( \text{Rel} = \text{Rel} \)]

by blast

thus \( SP\downarrow<\text{SWB}>a \)

by simp

qed

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-reflect-barbs:
fixes \( S\text{Rel} :: (\mathcal{S} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set
and \( T\text{Rel} :: (\mathcal{T} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set

assumes reflection: rel-reflects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)

shows rel-reflects-barbs SRel SWB
and rel-reflects-barbs TRel TWB

proof

show rel-reflects-barbs SRel SWB
using reflection rel-with-source-impl-SRel-reflects-barbs[where 
\( \text{Rel} = \text{indRelRSTPO} \text{SRel} \text{TRel} \text{ and SRel=SRel} \)]

by (simp add: indRelRSTPO.source)

next

show rel-reflects-barbs TRel TWB
using reflection rel-with-target-impl-TRel-reflects-barbs[where 
\( \text{Rel} = \text{indRelRSTPO} \text{SRel} \text{TRel} \text{ and TRel=TRel} \)]

by (simp add: indRelRSTPO.target)

qed

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-reflect-barbs:
fixes \( S\text{Rel} :: (\mathcal{S} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set
and \( T\text{Rel} :: (\mathcal{T} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set

assumes reflection: rel-reflects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)

shows rel-reflects-barbs SRel SWB
and rel-reflects-barbs TRel TWB

proof

show rel-reflects-barbs SRel SWB
using reflection rel-with-source-impl-SRel-reflects-barbs[where 
\( \text{Rel} = \text{indRelLSTPO} \text{SRel} \text{TRel} \text{ and SRel=SRel} \)]

by (simp add: indRelLSTPO.source)

next

show rel-reflects-barbs TRel TWB
using reflection rel-with-target-impl-TRel-reflects-barbs[where 
\( \text{Rel} = \text{indRelLSTPO} \text{SRel} \text{TRel} \text{ and TRel=TRel} \)]

by (simp add: indRelLSTPO.target)

qed

lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-reflect-barbs:
fixes \( S\text{Rel} :: (\mathcal{S} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set
and \( T\text{Rel} :: (\mathcal{T} \rightarrow (\mathcal{T} \times \mathcal{T})) \) set

assumes reflection: rel-reflects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)

shows rel-reflects-barbs SRel SWB
and rel-reflects-barbs TRel TWB

proof

show rel-reflects-barbs SRel SWB
using reflection rel-with-source-impl-SRel-reflects-barbs[where 
\( \text{Rel} = \text{indRelSTEQ} \text{SRel} \text{TRel} \text{ and SRel=SRel} \)]

by (simp add: indRelSTEQ.source)

next

show rel-reflects-barbs TRel TWB
using reflection rel-with-target-impl-TRel-reflects-barbs[where
\[ \text{Rel} = \text{indRelSTEQ} \quad \text{SRel} \quad \text{TRel} \quad \text{TRel} = \text{TRel} \]

by (simp add: \text{indRelSTEQ.target})

qed

**Lemma (in encoding-wrt-barbs)** \text{rel-with-source-impl-SRel-weakly-reflects-barbs}:

**Fixes** \text{SRel} :: `(procS × procS) set

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

**Assumes** reflection: rel-weakly-reflects-barbs \text{Rel} \text{(STCalWB SWB TWB)}

and \text{sourceInRel}: \forall S1 S2. (S1, S2) \in \text{SRel} \rightarrow (\text{SourceTerm S1, SourceTerm S2}) \in \text{Rel}

**Shows** rel-weakly-reflects-barbs \text{SRel SWB}

**Proof**:

clarify

fix \text{SP SQ a SQ}'

assume \((\text{SP}, \text{SQ}) \in \text{SRel})

with \text{sourceInRel} \text{(SourceTerm SP, SourceTerm SQ) \in \text{Rel}}

by blast

moreover assume \text{SQ} \rightarrow (\text{Calculus SWB})* \text{SQ}' and \text{SQ}'↓<\text{SWB}>a

hence \text{SourceTerm SQ}↓.a

by blast

ultimately have \text{SourceTerm SP}↓.a

by blast

thus \text{SP}↓<\text{SWB}>a

by simp

qed

**Lemma (in encoding-wrt-barbs)** \text{indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs}:

**Fixes** \text{SRel} :: `(procS × procS) set

and \text{TRel} :: `(procT × procT) set

**Assumes** reflection: rel-weakly-reflects-barbs \text{(indRelRSTPO SRel TRel) (STCalWB SWB TWB)}

**Shows** rel-weakly-reflects-barbs \text{SRel SWB}

and rel-weakly-reflects-barbs \text{TRel TWB}

**Proof**:

\(-\)

show rel-weakly-reflects-barbs \text{SRel SWB}

using reflection rel-with-source-impl-SRel-weakly-reflects-barbs[where \text{Rel} = \text{Rel}]

by (simp add: \text{indRelRSTPO.source})

next

show rel-weakly-reflects-barbs \text{TRel TWB}

using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where \text{Rel} = \text{Rel}]

by (simp add: \text{indRelRSTPO.target})

qed

**Lemma (in encoding-wrt-barbs)** \text{indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs}:

**Fixes** \text{SRel} :: `(procS × procS) set

and \text{TRel} :: `(procT × procT) set

**Assumes** reflection: rel-weakly-reflects-barbs \text{(indRelLSTPO SRel TRel) (STCalWB SWB TWB)}

**Shows** rel-weakly-reflects-barbs \text{SRel SWB}

and rel-weakly-reflects-barbs \text{TRel TWB}

**Proof**:

\(-\)

show rel-weakly-reflects-barbs \text{SRel SWB}

using reflection rel-with-source-impl-SRel-weakly-reflects-barbs[where \text{Rel} = \text{Rel}]

by (simp add: \text{indRelLSTPO.source})

next

show rel-weakly-reflects-barbs \text{TRel TWB}

using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where \text{Rel} = \text{Rel}]

by (simp add: \text{indRelLSTPO.target})

qed
lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-reflect-barbs:
fixes SRel :: (′procS × ′procS) set
and TRel :: (′procT × ′procT) set
assumes reflection: rel-weakly-reflects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
shows rel-weakly-reflects-barbs SRel SWB
and rel-weakly-reflects-barbs TRel TWB
proof −
  show rel-weakly-reflects-barbs SRel SWB
    using reflection rel-with-source-impl-SRel-weakly-reflects-barbs[where
      Rel=indRelSTEQ SRel TRel and SRel=SRel]
    by (simp add: indRelSTEQ.source)
next
  show rel-weakly-reflects-barbs TRel TWB
    using reflection rel-with-target-impl-TRel-weakly-reflects-barbs[where
      Rel=indRelSTEQ SRel TRel and TRel=TRel]
    by (simp add: indRelSTEQ.target)
qed

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

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-respect-barbs:
fixes SRel :: (′procS × ′procS) set
and TRel :: (′procT × ′procT) set
assumes respection: rel-respects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
shows rel-respects-barbs SRel SWB
and rel-respects-barbs TRel TWB
proof −
  show rel-respects-barbs SRel SWB
    using respection
      indRelRSTPO-impl-SRel-and-TRel-preserve-barbs[1][where SRel=SRel and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-reflect-barbs[1][where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
      indRelRSTPO-impl-SRel-and-TRel-preserve-barbs[2][where SRel=SRel and TRel=TRel]
      indRelRSTPO-impl-SRel-and-TRel-reflect-barbs[2][where SRel=SRel and TRel=TRel]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-respect-barbs:
fixes SRel :: (′procS × ′procS) set
and TRel :: (′procT × ′procT) set
assumes respection: rel-respects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
shows rel-respects-barbs SRel SWB
and rel-respects-barbs TRel TWB
proof −
  show rel-respects-barbs SRel SWB
    using respection
      indRelLSTPO-impl-SRel-and-TRel-preserve-barbs[1][where SRel=SRel and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-reflect-barbs[1][where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
      indRelLSTPO-impl-SRel-and-TRel-preserve-barbs[2][where SRel=SRel and TRel=TRel]
      indRelLSTPO-impl-SRel-and-TRel-reflect-barbs[2][where SRel=SRel and TRel=TRel]
    by blast
qed
lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-respect-barbs:
  fixes SRel :: (′procS × ′procS) set
  and TRel :: (′procT × ′procT) set
  assumes respection: rel-respects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)
  shows rel-respects-barbs SRel SWB
  and rel-respects-barbs TRel TWB
proof –
  show rel-respects-barbs SRel SWB
    using respection
    indRelSTEQ-impl-SRel-and-TRel-preserve-barbs(1) [where SRel=SRel and TRel=TRel]
    indRelSTEQ-impl-SRel-and-TRel-reflect-barbs(1) [where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-respects-barbs TRel TWB
    using respection
    indRelSTEQ-impl-SRel-and-TRel-preserve-barbs(2) [where SRel=SRel and TRel=TRel]
    indRelSTEQ-impl-SRel-and-TRel-reflect-barbs(2) [where SRel=SRel and TRel=TRel]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelRSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  fixes SRel :: (′procS × ′procS) set
  and TRel :: (′procT × ′procT) set
  assumes respection: rel-weakly-respects-barbs (indRelRSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-respects-barbs SRel SWB
  and rel-weakly-respects-barbs TRel TWB
proof –
  show rel-weakly-respects-barbs SRel SWB
    using respection
    indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(1) [where SRel=SRel and TRel=TRel]
    indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(1) [where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-weakly-respects-barbs TRel TWB
    using respection
    indRelRSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(2) [where SRel=SRel and TRel=TRel]
    indRelRSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(2) [where SRel=SRel and TRel=TRel]
    by blast
qed

lemma (in encoding-wrt-barbs) indRelLSTPO-impl-SRel-and-TRel-weakly-respect-barbs:
  fixes SRel :: (′procS × ′procS) set
  and TRel :: (′procT × ′procT) set
  assumes respection: rel-weakly-respects-barbs (indRelLSTPO SRel TRel) (STCalWB SWB TWB)
  shows rel-weakly-respects-barbs SRel SWB
  and rel-weakly-respects-barbs TRel TWB
proof –
  show rel-weakly-respects-barbs SRel SWB
    using respection
    indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(1) [where SRel=SRel and TRel=TRel]
    indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(1) [where SRel=SRel and TRel=TRel]
    by blast
next
  show rel-weakly-respects-barbs TRel TWB
    using respection
    indRelLSTPO-impl-SRel-and-TRel-weakly-preserve-barbs(2) [where SRel=SRel and TRel=TRel]
    indRelLSTPO-impl-SRel-and-TRel-weakly-reflect-barbs(2) [where SRel=SRel and TRel=TRel]
    by blast
qed
lemma (in encoding-wrt-barbs) indRelSTEQ-impl-SRel-and-TRel-weakly-respect-barbs:

fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set

assumes respection: rel-weakly-respects-barbs (indRelSTEQ SRel TRel) (STCalWB SWB TWB)

shows rel-weakly-respects-barbs SRel SWB
and rel-weakly-respects-barbs TRel TWB

proof –

show rel-weakly-respects-barbs SRel SWB
using respection indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs(1)[where SRel=SRel
and TRel=TRel]

by blast

next

show rel-weakly-respects-barbs TRel TWB
using respection indRelSTEQ-impl-SRel-and-TRel-weakly-preserve-barbs(2)[where SRel=SRel
and TRel=TRel]

by blast

qed

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

lemma (in encoding) indRelRTPO-to-indRelTEQ:

fixes TRel :: ('procT × 'procT) set
and P Q :: ('procS, 'procT) Proc

assumes rel: \( P \preceq_{\{\cdot\}} T <_{TRel} Q \)
and reflT: refl TRel

shows \( P \sim_{\{\cdot\}} T <_{TRel} Q \)

proof

induct

case (encR S)

show SourceTerm S \sim_{\{\cdot\}} T <_{TRel} TargetTerm (\[S\])
by (rule indRelTEQ.encR)

next

case (source S)

from reflT show SourceTerm S \sim_{\{\cdot\}} T <_{TRel} SourceTerm S

using indRelTEQ-refl[of TRel]

unfolding refl-on-def

by simp

next

case (target T1 T2)

assume (T1, T2) \in TRel

thus TargetTerm T1 \sim_{\{\cdot\}} T <_{TRel} TargetTerm T2

by (rule indRelTEQ.target)

next

case (trans TP TQ TR)

assume TP \sim_{\{\cdot\}} T <_{TRel} TQ and TQ \sim_{\{\cdot\}} T <_{TRel} TR

thus TP \sim_{\{\cdot\}} T <_{TRel} TR

by (rule indRelTEQ.trans)

qed

lemma (in encoding) indRelRTPO-to-indRelRSTPO:

fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and P Q :: ('procS, 'procT) Proc
assumes rel: $P \preceq \llbracket \cdot \rrbracket R < T Rel > Q$
and reflS: refl S Rel
shows $P \preceq \llbracket \cdot \rrbracket R < S Rel, T Rel > Q$
using rel

proof induct
  case (encR S)
  show SourceTerm S $\preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ TargetTerm (':[S])
    by (rule indRelRSTPO.encR)
next
  case (source S)
  from reflS show SourceTerm S $\preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ SourceTerm S
    unfolding refl-on-def
    by (simp add: indRelRSTPO.source)
next
  case (target T1 T2)
  assume (T1, T2) $\in$ T Rel
  thus TargetTerm T1 $\preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ TargetTerm T2
    by (rule indRelRSTPO.target)
next
  case (trans P Q R)
  assume $P \preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ Q and $Q \preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ R
  thus $P \preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ R
    by (rule indRelRSTPO.trans)
qed

lemma (in encoding) indRelRSTPO-to-indRelSTEQ:
  fixes S Rel :: (′procS × ′procS) set
  and T Rel :: (′procT × ′procT) set
  and P Q :: (′procS, ′procT) Proc
  assumes rel: $P \preceq \llbracket \cdot \rrbracket R < S Rel, T Rel >$ Q
  shows $P \sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ Q
  using rel
  proof induct
    case (encR S)
    show SourceTerm S $\sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ TargetTerm (':[S])
      by (rule indRelSTEQ.encR)
next
    case (source S1 S2)
    assume (S1, S2) $\in$ S Rel
    thus SourceTerm S1 $\sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ SourceTerm S2
      by (rule indRelSTEQ.source)
next
    case (target T1 T2)
    assume (T1, T2) $\in$ T Rel
    thus TargetTerm T1 $\sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ TargetTerm T2
      by (rule indRelSTEQ.target)
next
    case (trans P Q R)
    assume $P \sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ Q and $Q \sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ R
    thus $P \sim \llbracket \cdot \rrbracket R < S Rel, T Rel >$ R
      by (rule indRelSTEQ.trans)
  qed

If indRelRTPO is a bisimulation and S Rel is a reflexive bisimulation then also indRelRSTPO is a bisimulation.

lemma (in encoding) indRelRTPO-weak-reduction-bisimulation-impl-indRelRSTPO-bisimulation:
  fixes S Rel :: (′procS × ′procS) set
  and T Rel :: (′procT × ′procT) set
  assumes bisimT: weak-reduction-bisimulation (indRelRTPO T Rel) (STCal Source Target)
  and bisimS: weak-reduction-bisimulation S Rel Source
and reflS: refl SRel
shows weak-reduction-bisimulation (indRelRSTPO SRel TRel) (STCal Source Target)

proof auto

next

proof (induct arbitrary: P)

next

next

next

next

next

next

next

next

next

next

next

next
by blast

qed

next

fix $P \ Q \ Q'$
assume $P \leq \Gamma \ R_{<SRel,TRel>} \ Q$ and $Q \rightarrow (\text{SCal Source Target}) \ast \ Q'$
thus $\exists P'. \ P \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$

proof (induct arbitrary: $Q'$)

\begin{itemize}
    \item case (encR $S$)
        \begin{itemize}
            \item have $\text{SourceTerm } S \leq \Gamma \ [R_{<TRel>} \ \text{TargetTerm } (\{S\})]$
                by (rule indRelRTPO.encR)
        \end{itemize}
    \item moreover assume $\text{TargetTerm } (\{S\}) \rightarrow (\text{SCal Source Target}) \ast \ Q'$
    \item ultimately obtain $P'$ where $E1: \text{SourceTerm } S \rightarrow (\text{SCal Source Target}) \ast \ P'$
        and $E2: \ P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
        using bisimT
        by blast
    \item from refS E2 have $P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
        by (simp add: indRelRTPO-to-indRelRSTPO)
    \item with $E1$ show $\exists P'. \ \text{SourceTerm } S \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
        by blast
\end{itemize}

next

\begin{itemize}
    \item case (source $S1 \ S2$)
        \begin{itemize}
            \item assume $\text{SourceTerm } S2 \rightarrow (\text{SCal Source Target}) \ast \ Q'$
            \item from this obtain $S2' \ \text{where } F1: \ S2' \in S \ Q'$ and $F2: \ S2 \rightarrow \text{Source} \ast \ S2'$
                by (auto simp add: SCal-steps(1))
            \item assume $(S1, S2) \in SRel$
                with $F2$ bisimS obtain $S1' \ \text{where } F3: \ S1 \rightarrow \text{Source} \ast \ S1'$ and $F4: \ (S1', S2') \in SRel$
                by blast
            \item from $F3$ have $\text{SourceTerm } S1 \rightarrow (\text{SCal Source Target}) \ast (\text{SourceTerm } S1')$
                by (simp add: SCal-steps(1))
            \item moreover from $F1 \ F4$ have $\text{SourceTerm } S1' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                by (simp add: indRelRSTPO.source)
            \item ultimately show $\exists P'. \ \text{SourceTerm } S1 \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                by blast
        \end{itemize}
    \item case (target $T1 \ T2$)
        \begin{itemize}
            \item assume $(T1, T2) \in TRel$
                hence $\text{TargetTerm } T1 \leq \Gamma \ R_{<TRel>} \ \text{TargetTerm } T2$
                by (rule indRelRTPO.target)
            \item moreover assume $\text{TargetTerm } T2 \rightarrow (\text{SCal Source Target}) \ast \ Q'$
            \item ultimately obtain $P'$ where $G1: \ \text{TargetTerm } T1 \rightarrow (\text{SCal Source Target}) \ast \ P'$
                and $G2: \ P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                using bisimT
                by blast
            \item from refS G2 have $P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                by (simp add: indRelRTPO-to-indRelRSTPO)
            \item with $G1$ show $\exists P'. \ \text{TargetTerm } T1 \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                by blast
        \end{itemize}
    \item case (trans $P \ Q \ R \ R'$)
        \begin{itemize}
            \item assume $R \rightarrow (\text{SCal Source Target}) \ast \ R'$
                and $\bigwedge \ R'. \ R \rightarrow (\text{SCal Source Target}) \ast \ R'$
                    \[ \Rightarrow \exists Q'. \ Q \rightarrow (\text{SCal Source Target}) \ast \ Q' \land Q' \leq \Gamma \ R_{<SRel,TRel>} \ R' \]
            \item from this obtain $Q'$ where $H1: \ Q \rightarrow (\text{SCal Source Target}) \ast \ Q'$ and $H2: \ Q' \leq \Gamma \ R_{<SRel,TRel>} \ R'$
                by blast
            \item assume $\bigwedge \ Q'. \ Q \rightarrow (\text{SCal Source Target}) \ast \ Q'$
                    \[ \Rightarrow \exists P'. \ P \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ Q' \]
            \item with $H1$ obtain $P'$ where $H3: \ P \rightarrow (\text{SCal Source Target}) \ast \ P'$ and $H4: \ P' \leq \Gamma \ R_{<SRel,TRel>} \ Q'$
                by blast
            \item from $H4 \ H2$ have $P' \leq \Gamma \ R_{<SRel,TRel>} \ R'$
                by (rule indRelRSTPO.trans)
            \item with $H3$ show $\exists P'. \ P \rightarrow (\text{SCal Source Target}) \ast \ P' \land P' \leq \Gamma \ R_{<SRel,TRel>} \ R'$
        \end{itemize}
\end{itemize}
6 Success Sensitiveness and Barbs

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

abbreviation (in encoding-wrt-barbs) enc-weakly-preserves-barb-set :: 'barbs set ⇒ bool where
enc-weakly-preserves-barb-set Barbs ≡ enc-preserves-binary-pred (λP a. $\mathsf{a} \in$ Barbs ∧ $\mathsf{P} \downarrow a$)

abbreviation (in encoding-wrt-barbs) enc-weakly-preserves-barbs :: bool where
enc-weakly-preserves-barbs ≡ enc-preserves-binary-pred (λP a. $\mathsf{P} \downarrow a$)

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

abbreviation (in encoding-wrt-barbs) enc-weakly-reflects-barb-set :: 'barbs set ⇒ bool where
enc-weakly-reflects-barb-set Barbs ≡ enc-reflects-binary-pred (λP a. $\mathsf{a} \in$ Barbs ∧ $\mathsf{P} \downarrow a$)

abbreviation (in encoding-wrt-barbs) enc-weakly-reflects-barbs :: bool where
enc-weakly-reflects-barbs ≡ enc-reflects-binary-pred (λP a. $\mathsf{P} \downarrow a$)

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

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

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

lemma (in encoding-wrt-barbs) enc-weakly-respects-barbs-and-barb-set:
shows enc-weakly-respects-barbs = (∀ Barbs. enc-weakly-respects-barb-set Barbs)
proof
have (∀ Barbs. enc-weakly-respects-barb-set Barbs)
  = (∀ Barbs. (∀ S x. x ∈ Barbs ∧ $\mathsf{S} \downarrow SWB > x$ → $\mathsf{S} \downarrow TWB > x$)
  ∧ (∀ S x. x ∈ Barbs ∧ $\mathsf{S} \downarrow TWB > x$ → $\mathsf{S} \downarrow SWB > x$))
  by simp
hence (∀ Barbs. enc-weakly-respects-barb-set Barbs)
  = (∀ Barbs. enc-weakly-preserves-barb-set Barbs)
  ∧ (∀ Barbs. enc-weakly-reflects-barb-set Barbs))
apply simp by fast
thus ?thesis
An encoding strongly respects some set of barbs if, for every source term S and all considered barbs a, S has a iff the translation of S has a.

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

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

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

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

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

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

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

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

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

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

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

**lemma** (in encoding-wrt-barbs) enc-weakly-preserves-barbs-iff-source-target-rel:
shows enc-weakly-preserves-barbs
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])] ∈ Rel)
      ∧ rel-weakly-preserves-barbs Rel (STCalWB SWB TWB))
An encoding (weakly) respects barbs iff (1) there exists a relation, like \( \text{indRelR} \), that relates source terms and their literal translations and preserves (reachability/existence of barbs), or (2) there exists a relation, like \( \text{indRelL} \), that relates literal translations and their source terms and reflects (reachability/existence of barbs). An encoding (weakly) reflects barbs iff (1) there exists a relation, like \( \text{indRelR} \), that relates source terms and their literal translations and preserves (reachability/existence of barbs), or (2) there exists a relation, like \( \text{indRelL} \), that relates literal translations and their source terms and preserves (reachability/existence of barbs).

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

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

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

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

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

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

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

lemma (in encoding-wrt-barbs) success-sensitive-cond:
  fixes success :: 'barbs
  shows enc-weakly-respects-barb-set {success} = (∀ S. S↓<SWB>success ↔ [S]↓<TWB>success)
  by auto

lemma (in encoding-wrt-barbs) success-sensitive-iff-source-target-rel-weakly-respects-success:
  fixes success :: 'barbs
  shows enc-weakly-respects-barb-set {success}
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
   ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success})
  by (rule enc-weakly-respects-barb-set-iff-source-target-rel[where Barbs={success}])+

lemma (in encoding-wrt-barbs) success-sensitive-iff-source-target-rel-respects-success:
  fixes success :: 'barbs
  shows enc-respects-barb-set {success}
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
   ∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success})
  by (rule enc-respects-barb-set-iff-source-target-rel[where Barbs={success}])
theory DivergenceReflection
imports SourceTargetRelation
begin

7 Divergence Reflection

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

abbreviation (in encoding) enc-preserves-divergence :: bool where
enc-preserves-divergence ≡ enc-preserves-pred (λP. P →→ STω)

lemma (in encoding) divergence-preservation-cond:
shows enc-preserves-divergence = (∀ S. S →→ (Source)ω → [S] →→ (Target)ω)
by simp

abbreviation (in encoding) enc-reflects-divergence :: bool where
enc-reflects-divergence ≡ enc-reflects-pred (λP. P →→ STω)

lemma (in encoding) divergence-reflection-cond:
shows enc-reflects-divergence = (∀ S. [S] →→ (Target)ω → S →→ (Source)ω)
by simp

abbreviation rel-preserves-divergence :: (′proc × ′proc) set ⇒ ′proc processCalculus ⇒ bool
where
rel-preserves-divergence Rel Cal ≡ rel-preserves-pred Rel (λP. P →→ (Cal)ω)

abbreviation rel-reflects-divergence :: (′proc × ′proc) set ⇒ ′proc processCalculus ⇒ bool
where
rel-reflects-divergence Rel Cal ≡ rel-reflects-pred Rel (λP. P →→ (Cal)ω)

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

abbreviation (in encoding) enc-respects-divergence :: bool where
enc-respects-divergence ≡ enc-respects-pred (λP. P →→ STω)

lemma (in encoding) divergence-respection-cond:
shows enc-respects-divergence = (∀ S. [S] →→ (Target)ω ←→ S →→ (Source)ω)
by auto

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

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

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

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

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

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

An encoding respects divergence iff there exists a relation that relates source terms and their literal translations in both directions and respects divergence.

lemma (in encoding) divergence-respection-iff-source-target-rel-respects-divergence:
  shows enc-respects-divergence
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
  and enc-respects-divergence
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
  proof –
  show enc-respects-divergence
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
  using enc-respects-pred-iff-source-target-rel-respects-pred-encR[where Pred=λP. P ⟷ STω]
divergentST-STCal-divergent
  by simp
  next
  show enc-respects-divergence
  = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel ∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel)
       ∧ rel-respects-divergence Rel (STCal Source Target))
  using enc-respects-pred-iff-source-target-rel-respects-pred-encRL[where Pred=λP. P ⟷ STω]
divergentST-STCal-divergent
  by simp
  qed

end

theory OperationalCorrespondence
  imports SourceTargetRelation
begin
8 Operational Correspondence

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

abbreviation (in encoding) operational-complete :: ('procT × 'procT) set ⇒ bool where
operational-complete TRel ≡ ∀ S S'. S → Source S' → (∃ T. [S] → Target T ∧ ([S'], T) ∈ TRel)

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

abbreviation (in encoding) strongly-operational-complete :: ('procT × 'procT) set ⇒ bool where
strongly-operational-complete TRel ≡ ∀ S S'. S → Source S' → (∃ T. [S] → Target T ∧ ([S'], T) ∈ TRel)

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

abbreviation (in encoding) weakly-operational-sound :: ('procT × 'procT) set ⇒ bool where
weakly-operational-sound TRel ≡ ∀ S T. [S] → Target T → (∃ T'. S → Source S' ∧ T → Target T' ∧ ([S'], T) ∈ TRel)

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

abbreviation (in encoding) operational-sound :: ('procT × 'procT) set ⇒ bool where
operational-sound TRel ≡ ∀ S T. [S] → Target T → (∃ T'. S → Source S' ∧ ([S'], T) ∈ TRel)

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

abbreviation (in encoding) strongly-operational-sound :: ('procT × 'procT) set ⇒ bool where
strongly-operational-sound TRel ≡ ∀ S T. [S] → Target T → (∃ T'. S → Source S' ∧ ([S'], T) ∈ TRel)

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

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

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

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

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

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

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

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

- shows operational-complete \{\{T_1, T_2\}. True\}
- and weakly-operational-sound \{\{T_1, T_2\}. True\}
- and operational-sound \{\{T_1, T_2\}. True\}
  
  using steps-refl[where Cal=Source] steps-refl[where Cal=Target]
  
  by blast+

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

- fixes Cal :: 'a processCalculus
- shows weak-reduction-bisimulation \{\{a, b\}. True\} Cal
  
  using steps-refl[where Cal=Cal]
  
  by blast

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

- shows \(\exists T_{Rel}. \text{weakly-operational-corresponding } T_{Rel}\)
- and \(\exists T_{Rel}. \text{operational-corresponding } T_{Rel}\)
  
  using \(\text{operational-correspondence-modulo-all-relation}\)
  
  by blast+

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

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

- shows \(\exists T_{Rel}. \text{strongly-operational-corresponding } T_{Rel}\)
  
  using \(\text{steps-refl}\)
  
  by blast

- assumes \(\exists T_{Rel}. \text{strongly-operational-corresponding } T_{Rel}\)
  
  then \(\forall S. (\exists S'. S \mapsto \text{Source } S') \iff (\exists T. [S] \mapsto \text{Target } T)\)
  
  using \(\text{weak-reduction-bisimulation}\)
  
  by blast

- assumes \(\exists T_{Rel}. \text{strongly-operational-corresponding } T_{Rel}\)
  
  then \(\forall S. (\exists S'. S \mapsto \text{Source } S') \iff (\exists T. [S] \mapsto \text{Target } T)\)
  
  using \(\text{all-relation-is-weak-reduction-bisimulation}\)
  
  by blast

- assumes \(\exists T_{Rel}. \text{strongly-operational-corresponding } T_{Rel}\)
  
  then \(\forall S. (\exists S'. S \mapsto \text{Source } S') \iff (\exists T. [S] \mapsto \text{Target } T)\)
  
  by blast

- assumes \(\exists T_{Rel}. \text{strongly-operational-corresponding } T_{Rel}\)
  
  then \(\forall S. (\exists S'. S \mapsto \text{Source } S') \iff (\exists T. [S] \mapsto \text{Target } T)\)
  
  by blast

qed
8.2 (Strong) Operational Completeness vs (Strong) Simulation

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

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

```
fixes Rel :: (('procS', 'procT) Proc × ('procS', 'procT) Proc) set
and TRel :: ('procT × 'procT) set
assumes A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and A2: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⇒ ([S], T) ∈ TRel
and A3: weak-reduction-simulation Rel (STCal Source Target)
shows operational-complete (TRel)
```

**proof clarify**
- fix \( S S' \)
- from A1 have (SourceTerm S, TargetTerm ([S])) ∈ Rel
- by simp
- moreover assume \( S \mapsto Source\ast S' \)
- hence SourceTerm S ⇒ (STCal Source Target)\ast (SourceTerm S')
- by (simp add: STCal-steps(1))
- ultimately obtain \( Q' \) where A5: TargetTerm ([S]) ⇒ (STCal Source Target)\ast Q'
- and A6: (SourceTerm S', Q') ∈ Rel
- using A3
- by blast
- from A5 obtain T where A7: \( T ∈ T Q' \) and A8: ([S]) ⇒ Target\ast T
- by (auto simp add: STCal-steps(2))
- from A2 A6 A7 have ([S], T) ∈ TRel
- by simp
- with A8 show \( ∃ T. ([S] ⇒ Target\ast T ∧ ([S'], T) ∈ TRel) \)
- by blast

**QED**

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

```
fixes TRel :: ('procT × 'procT) set
shows (operational-complete (TRel))
  ∧ weak-reduction-simulation (TRel\ast Target)
  ⇒ weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
```

**proof clarify**
- fix \( P Q P' \)
- assume \( P ≤\{\}RT<TRel> Q \) and \( P ⇒(STCal Source Target)\ast P' \)
- thus \( ∃ Q'. Q ⇒(STCal Source Target)\ast Q' ∧ P' ≤\{\}RT<TRel> Q' \)
- **proof** (induct arbitrary: \( P' \))
- **case (encR S)**
- assume SourceTerm S ⇒(STCal Source Target)\ast P'
- from this obtain \( S' \) where A1: \( S' ∈ S \) \( P' \) and A2: \( S ⇒ Source\ast S' \)
- by (auto simp add: STCal-steps(1))
- from oc A2 obtain T where A3: \( [S] ⇒ Target\ast T \) and A4: \( ([S'], T) ∈ TRel \)
- by blast
- from A3 have TargetTerm ([S]) ⇒(STCal Source Target)\ast (TargetTerm T)
- by (simp add: STCal-steps(2))
- moreover have \( P' ≤\{\}RT<TRel> TargetTerm T \)
- **proof**
- from A4 have \( [S'] = T ∨ ([S'], T) ∈ TRel \)
- using rtrancl-eq-or-rtrancl[of \( [S'] \) \( T \) \( TRel) \]
- by blast
- moreover from A1 have A5: \( P' ≤\{\}RT<TRel> TargetTerm ([S']) \)
- by (simp add: indRelRTPO.encR)
hence $[S'] = T \iff P' \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T$

by simp

moreover have $([S'], T) \in \text{TRel}^+ \iff P' \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T$

proof –

assume $([S'], T) \in \text{TRel}^+$

hence $\text{TargetTerm} ([S']) \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T$

proof induct

fix $T$

assume $([S'], T) \in \text{TRel}^+$

thus $\text{TargetTerm} ([S']) \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T$

by (rule indRelRTPO.target)

next

case (step TQ TR)

assume $\text{SourceTerm} S \mapsto (\text{STCal Source Target}) \ast P'$

moreover from this obtain $S' \text{ where } B1: S' \in S P'$

by (auto simp add: STCal-steps(1))

hence $P' \leq [\cdot RT < \text{TRel}] \Rightarrow P'$

by (simp add: indRelRTPO.source)

ultimately show $\exists Q'. \text{SourceTerm} S \mapsto (\text{STCal Source Target}) \ast Q' \land P' \leq [\cdot RT < \text{TRel}] \Rightarrow Q'$

by blast

next

case (source $S$)

assume $\text{TargetTerm} T1 \mapsto (\text{STCal Source Target}) \ast P'$

moreover from this obtain $T1' \text{ where } C1: T1' \in T P' \land C2: T1 \mapsto \text{Target} \ast T1'$

by (auto simp add: STCal-steps(2))

assume $(T1, T2) \in \text{TRel}^+$

hence $(T1, T2) \in \text{TRel}^+$

by simp

with $C2 \text{ sim obtain } T2' \text{ where } C3: T2 \mapsto \text{Target} \ast T2'$

and $C4: (T1', T2') \in \text{TRel}^+$

by blast

from $C3$ have $\text{TargetTerm} T2 \mapsto (\text{STCal Source Target}) \ast (\text{TargetTerm} T2')$

by (simp add: STCal-steps(2))

moreover from $C4$ have $\text{TargetTerm} T1' \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T2'$

proof induct

fix $T2'$

assume $(T1', T2') \in \text{TRel}^+$

thus $\text{TargetTerm} T1' \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} T2'$

by (rule indRelRTPO.target)

next

case (step TQ TR)

assume $\text{TargetTerm} T1' \leq [\cdot RT < \text{TRel}] \Rightarrow \text{TargetTerm} TQ$

moreover assume $(TQ, TR) \in \text{TRel}$
[Proof]

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

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

**Shows** (operational-complete (TRel+)) and weak-reduction-simulation (TRel+) Target

**Proof** (rule iff, erule conjE)

have $\exists S. \text{SourceTerm } S \leq [\cdot]\text{RT}<\text{TRel} > \text{TargetTerm } ([S])$

by (rule add: indRelRTPO.encR)

moreover have $\forall S T. \text{SourceTerm } S \leq [\cdot]\text{RT}<\text{TRel} > \text{TargetTerm } T \rightarrow ([S], T) \in \text{TRel}^*$

using indRelRTPO-to-TRel(2)[where $\text{TRel} = \text{TRel}^*$ trans-closure-of-TRel-refl-cond]

by simp

moreover assume sim: weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)

ultimately have operational-complete (TRel+)

using weak-reduction-simulation-impl-OCom[where $\text{Rel} = \text{indRelRTPO TRel}$ and $\text{TRel} = \text{TRel}^*$]

by simp

moreover from sim have weak-reduction-simulation (TRel+) Target

using indRelRTPO-impl-TRel-is-weak-reduction-simulation[where $\text{TRel} = \text{TRel}^*$]

by simp

ultimately show operational-complete (TRel+)

and weak-reduction-simulation (TRel+) Target

by simp

qed
by simp

moreover have \( \forall S \ T. \ SourceTerm S \leq_{\text{RTPO}} T \) \( \Rightarrow \) \( \TargetTerm T \rightarrow ([S], T) \in TRel^* \)

using \( \text{indRelRTPO-to-TRel}(2) \) [where \( TRel=TRel \)] trans-closure-of-TRel-refl-cond

by simp

moreover assume operational-complete (TRel*)

and weak-reduction-simulation (TRel*) Target

hence weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)

using OCom-iff-indRelRTPO-is-weak-reduction-simulation [where \( TRel=TRel \)]

by simp

ultimately show \( \exists Rel. \ (\forall S. \ (SourceTerm S, TargetTerm ([S])) \in Rel) \)

\( \land \ (\forall T1 T2. \ (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \)

\( \land \ (\forall T1 T2. \ (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+) \)

\( \land \ (\forall S T. \ (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^*) \)

\( \land \ weak-reduction-simulation Rel \ (STCal Source Target) \)

by blast

next

assume \( \exists Rel. \ (\forall S. \ (SourceTerm S, TargetTerm ([S])) \in Rel) \)

\( \land \ (\forall T1 T2. \ (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \)

\( \land \ (\forall T1 T2. \ (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+) \)

from this obtain \( Rel \ where \ A1: \ (\forall S. \ (SourceTerm S, TargetTerm ([S])) \in Rel) \)

and \( A2: \ (\forall T1 T2. \ (T1, T2) \in TRel \rightarrow (TargetTerm T1, TargetTerm T2) \in Rel) \)

and \( A3: \ (\forall T1 T2. \ (TargetTerm T1, TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+) \)

and \( A4: \ (\forall S T. \ (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^*) \)

and \( A5: \ weak-reduction-simulation Rel \ (STCal Source Target) \)

by blast

from \( A1 \ A4 \ A5 \) have operational-complete (TRel*)

using weak-reduction-simulation-impl-OCm [where \( Rel=Rel \) and \( TRel=TRel \)]

by simp

moreover from \( A2 \ A3 \ A5 \) have weak-reduction-simulation (TRel*) Target

using rel-with-target-impl-transC-TRel-is-weak-reduction-simulation [where \( Rel=Rel \) and \( TRel=TRel \)]

by simp

ultimately show operational-complete (TRel*)

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

by simp

qed

An encoding is strong operational complete modulo a strong simulation on target terms TRel if 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 \( Rel := ((\text{proc}S, \text{proc}T) \ Proc \times (\text{proc}S, \text{proc}T) \ Proc) \) set

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

assumes \( A1: \ (\forall S. \ (SourceTerm S, TargetTerm ([S])) \in Rel \)

and \( A2: \ (\forall S T. \ (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^*) \)

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

shows strongly-operational-complete (TRel*)

proof clarify

fix \( S \ S' \)

from \( A1 \) have \( (SourceTerm S, TargetTerm ([S])) \in Rel \)

by simp

moreover assume \( S \mapsto Source S' \)

hence \( SourceTerm S \mapsto (STCal Source Target) \) \( (SourceTerm S') \)

by (simp add: STCal-step(1))

ultimately obtain \( Q' \) where \( A5: \ TargetTerm ([S]) \mapsto (STCal Source Target) \) \( Q' \)

and \( A6: \ (SourceTerm S', Q') \in Rel \)

using \( A3 \)
by blast
from A5 obtain T where A7: T ∈ T Q′ and A8: [S] ⊢ Target T
  by (auto simp add: STCal-step(2))
from A2 A6 A7 have ([S′], T) ∈ TRel+
  by simp
with A8 show ∃ T. [S] ⊢ Target T ∧ ([S′], T) ∈ TRel+
  by blast
qed

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

qed
ultimately

show \( \exists Q'. TargetTerm ([S]) \rightarrow (STCal Source Target) Q' \land P' \preceq RT \langle T\rangle RT \rightarrow Q' \)
by blast

next

case (source S)
assume SourceTerm S \rightarrow (STCal Source Target) P'
moreover from this obtain S' where B1: S' \in S P'
by (auto simp add: STCal-step(1))

hence \( P' \preceq RT \langle T\rangle RT \rightarrow P' \)
by (simp add: indRelRTPO.source)

ultimately show \( \exists Q'. SourceTerm S \rightarrow (STCal Source Target) Q' \land P' \preceq RT \langle T\rangle RT \rightarrow Q' \)
by blast

next

case (target T1 T2)
assume TargetTerm T1 \rightarrow (STCal Source Target) P'
from this obtain T' where C1: T1' \in T P' and C2: T1 \rightarrow Target T1'
by (auto simp add: STCal-step(2))

assume (T1, T2) \in TRel

hence (T1, T2) \in TRel+
by simp

with C2 sim obtain T2' where C3: T2 \rightarrow Target T2' and C4: (T1', T2') \in TRel+
by blast

from C3 have TargetTerm T2 \rightarrow (STCal Source Target) (TargetTerm T2')
by (simp add: STCal-step(2))

moreover from C4 have TargetTerm T1' \preceq RT \langle T\rangle TargetTerm T2'

proof induct

fix T2'

assume (T1', T2') \in TRel

thus TargetTerm T1' \preceq RT \langle T\rangle TargetTerm T2'
by (rule indRelRTPO.backward)

next

case (step TQ TR)
assume TargetTerm T1' \preceq RT \langle T\rangle TargetTerm TQ
moreover assume (TQ, TR) \in TRel

hence TargetTerm TQ \preceq RT \langle T\rangle TargetTerm TR
by (rule indRelRTPO.backward)

ultimately show TargetTerm T1' \preceq RT \langle T\rangle TargetTerm TR
by (rule indRelRTPO.backward)

qed

with C1 have P' \preceq RT \langle T\rangle TargetTerm T2'
by simp

ultimately show \( \exists Q'. TargetTerm T2 \rightarrow (STCal Source Target) Q' \land P' \preceq RT \langle T\rangle RT \rightarrow Q' \)
by blast

next

case (trans P Q R)
assume P \rightarrow (STCal Source Target) P'
and \( \lambda P'. P \rightarrow (STCal Source Target) P' \)
\implies \( \exists Q'. Q \rightarrow (STCal Source Target) Q' \land P' \preceq RT \langle T\rangle RT \rightarrow Q' \)

from this obtain Q' where D1: Q \rightarrow (STCal Source Target) Q'
and D2: P' \preceq RT \langle T\rangle RT \rightarrow Q'
by blast

assume \( \lambda Q'. Q \rightarrow (STCal Source Target) Q' \)
\implies \( \exists Q'. R \rightarrow (STCal Source Target) R' \land Q' \preceq RT \langle T\rangle RT \rightarrow R' \)

with D1 obtain R' where D3: R \rightarrow (STCal Source Target) R'
and D4: Q' \preceq RT \langle T\rangle RT \rightarrow R'
by blast

from D2 D4 have P' \preceq RT \langle T\rangle RT \rightarrow R'
by (rule indRelRTPO.backward)
with \( D3 \) show \( \exists R'. R \mapsto (STCal \ Source \ Target) R' \land P' \leq R \mapsto TRel > R' \)
by blast
qed
qed

next
have \( \forall S. \ SourceTerm S \leq R \mapsto TRel > \ TargetTerm ([S]) \)
by \( \text{simp add: indRelRTPO.encR} \)
moreover have \( \forall S T. \ SourceTerm S \leq R \mapsto TRel > \ TargetTerm T \mapsto ([S], T) \in TRel^* \)
using \( \text{indRelRTPO-to-TRel\_\_1 \ where TRel=TRel \ trans\_\_closure\_\_of\_\_TRel\_\_refl\_\_cond} \)
by simp
moreover assume \( \text{sim: strong\_\_reduction\_\_simulation (indRelRTPO TRel) (STCal Source Target)} \)
ultimately have \( \text{strong\_\_operation\_\_complete (TRel^*)} \)
using \( \text{strong\_\_reduction\_\_simulation\_\_impl\_SOCom \ where TRel=TRel \ and TRel=TRel} \)
by simp
moreover from \( \text{sim have strong\_\_reduction\_\_simulation (TRel^*) Target} \)
using \( \text{indRelRTPO\_\_impl\_\_TRel\_\_is\_\_strong\_\_reduction\_\_simulation \ where TRel=TRel} \)
by simp
ultimately show \( \text{strong\_\_operation\_\_complete (TRel^*)} \)
\& \( \text{strong\_\_reduction\_\_simulation (TRel^*)} \)
by simp
qed

lemma \( \text{(in encoding) SOCom\_iff\_strong\_\_reduction\_\_simulation:} \)
fixes TRel :: \( \langle \text{procT} \times \text{procT} \rangle \) set
shows \( \text{(strong\_\_operation\_\_complete (TRel^*)} \)
\& \( \text{strong\_\_reduction\_\_simulation (TRel^*) Target} \)
\( = (\exists \ Rel. \ (\forall S. \ (\text{SourceTerm S}, \ \text{TargetTerm ([S]})) \in \ Rel) \)
\& \( (\forall T1 T2. \ (T1, T2) \in TRel \mapsto (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in TRel) \)
\& \( (\forall T1 T2. \ (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in TRel \mapsto (T1, T2) \in TRel^+) \)
\& \( (\forall S T. \ (\text{SourceTerm S}, \ \text{TargetTerm T}) \in Rel \mapsto ([S], T) \in TRel^+) \)
\& \( \text{strong\_\_reduction\_\_simulation Rel (STCal Source Target)} \)
proof \( \text{(rule iffI, erule conjE)} \)
have \( \forall S. \ (\text{SourceTerm S}, \ \text{TargetTerm ([S]})) \in \ indRelRTPO TRel \)
by \( \text{simp add: indRelRTPO.encR} \)
moreover have \( \forall T1 T2. \ (T1, T2) \in TRel \mapsto \text{TargetTerm T1} \leq R \mapsto TRel > \text{TargetTerm T2} \)
by \( \text{simp add: indRelRTPO.target} \)
moreover have \( \forall T1 T2. \ (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in TRel \mapsto (T1, T2) \in TRel^+ \)
using \( \text{indRelRTPO-to-TRel\_\_4 \ where TRel=TRel} \)
by simp
moreover have \( \forall S T. \ (\text{SourceTerm S}, \ \text{TargetTerm T}) \in Rel \mapsto ([S], T) \in TRel^+ \)
using \( \text{indRelRTPO-to-TRel\_\_2 \ where TRel=TRel \ trans\_\_closure\_\_of\_\_TRel\_\_refl\_\_cond} \)
by simp
moreover assume \( \text{strong\_\_operation\_\_complete (TRel^*)} \)
and \( \text{strong\_\_reduction\_\_simulation (TRel^*) Target} \)
using \( \text{SOCom\_iff\_indRelRTPO TRel \ is\_\_strong\_\_reduction\_\_simulation \ where TRel=TRel} \)
by simp
ultimately show \( \exists \ Rel. \ (\forall S. \ (\text{SourceTerm S}, \ \text{TargetTerm ([S]})) \in \ Rel) \)
\& \( (\forall T1 T2. \ (T1, T2) \in TRel \mapsto (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in Rel) \)
\& \( (\forall T1 T2. \ (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in Rel \mapsto (T1, T2) \in TRel^+) \)
\& \( (\forall S T. \ (\text{SourceTerm S}, \ \text{TargetTerm T}) \in Rel \mapsto ([S], T) \in TRel^+) \)
\& \( \text{strong\_\_reduction\_\_simulation Rel (STCal Source Target)} \)
by blast
next
assume \( \exists \ Rel. \ (\forall S. \ (\text{SourceTerm S}, \ \text{TargetTerm ([S]})) \in \ Rel) \)
\& \( (\forall T1 T2. \ (T1, T2) \in TRel \mapsto (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in Rel) \)
\& \( (\forall T1 T2. \ (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in Rel \mapsto (T1, T2) \in TRel^+) \)
\& \( (\forall S T. \ (\text{SourceTerm S}, \ \text{TargetTerm T}) \in Rel \mapsto ([S], T) \in TRel^+) \)
\& \( \text{strong\_\_reduction\_\_simulation Rel (STCal Source Target)} \)
from \( \text{this obtain Rel where A1:} \forall S. \ (\text{SourceTerm S}, \ \text{TargetTerm ([S]})) \in \ Rel \)
and \( A2: \forall T1 T2. \ (T1, T2) \in TRel \mapsto (\text{TargetTerm T1}, \ \text{TargetTerm T2}) \in Rel \)
and A3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel+
and A4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*
and A5: strongly-reduction-simulation Rel (STCal Source Target)
by blast
from A1 A4 A5 have strongly-operational-complete (TRel*)
  using strongly-reduction-simulation-impl-SOCom[where Rel=Rel and TRel=TRel]
by simp
moreover from A2 A3 A5 have strong-reduction-simulation (TRel+) Target
  using rel-with-target-impl-transC-TRel-is-strong-reduction-simulation[where Rel=Rel and TRel=TRel]
by simp
ultimately show strongly-operational-complete (TRel*)
  ∧ strongly-reduction-simulation (TRel+) Target
  by simp
qed

lemma (in encoding) target-relation-from-source-target-relation:
  fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  assumes stre: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel
           → (TargetTerm ([S]), TargetTerm T) ∈ Rel=
  shows ∃ TRel. (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
        ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → (T1, T2) ∈ TRel+)
        ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*)
proof –
def trel: TRel≡{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
from trel have ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
  by simp
moreover from trel have ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ TRel → (T1, T2) ∈ TRel+
  by blast
moreover from stre trel have ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel*
  by blast
ultimately show ?thesis
  by blast
qed

lemma (in encoding) SOCom-modulo-TRel-iff-strong-reduction-simulation:
  shows (∃ TRel. strongly-operational-complete (TRel*)
           ∧ strongly-reduction-simulation (TRel+) Target)
        = (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
            ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → (TargetTerm ([S]), TargetTerm T) ∈ Rel=)
            ∧ strongly-reduction-simulation Rel (STCal Source Target))
proof (rule iffI)
  assume ∃ TRel. strongly-operational-complete (TRel*)
           ∧ strongly-reduction-simulation (TRel+) Target
  from this obtain TRel where strongly-operational-complete (TRel*)
  and strongly-reduction-simulation (TRel+) Target
  by blast
  hence strongly-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
    using SOCom-iff-indRelRTPO-is-strong-reduction-simulation[where TRel=TRel]
    by simp
  moreover have ∀ S. SourceTerm S ≲ [S] TargetTerm ([S])
    by (simp add: indRelRTPO_encR)
  moreover have ∀ S T. SourceTerm S ≲ [S] TargetTerm T
           → (TargetTerm ([S]), TargetTerm T) ∈ (indRelRTPO TRel)=
    using indRelRTPO-relates-source-target[where TRel=TRel]
    by simp
  ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
           ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel
            → (TargetTerm ([S]), TargetTerm T) ∈ Rel=)

\(\land\) strong-reduction-simulation \(\text{Rel}\) (\(\text{STCal Source Target}\))

\begin{itemize}
  \item by \textit{blast}
  \item next
  \item assume \(\exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\)
  \(\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}^{-1})\)
  \(\land\) strong-reduction-simulation \(\text{Rel}\) (\(\text{STCal Source Target}\))

\textbf{from this obtain} \(\text{Rel where } A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
  \(\land A2: (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}^{-1})\)
  \(\land A3: \text{strong-reduction-simulation } \text{Rel} (\text{STCal Source Target})\)

\end{itemize}

\begin{itemize}
  \item by \textit{blast}
  \item from \(A2\) obtain \(\text{TRel where } \forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\)
    \(\land \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^{+}\)
    \(\land \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^{*}\)

\item using \(\text{target-relation-from-source-target-relation[where } \text{Rel}=\text{Rel]}\)
  \begin{itemize}
    \item by \textit{blast}
    \item with \(A1 \ A3\) have \(\text{strongly-operational-complete } (\text{TRel}^{*})\)
      \(\land\) strong-reduction-simulation \(\text{TRel}^{+}\) \(\text{Target}\)

  \end{itemize}

\item thus \(\exists \text{TRel. strongly-operational-complete } (\text{TRel}^{*})\)
  \(\land\) strong-reduction-simulation \(\text{TRel}^{+}\) \(\text{Target}\)

\end{itemize}

\textbf{by} \textit{blast}

\textbf{qed}

### 8.3 Weak Operational Soundness vs Contrasimulation

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

\textbf{lemma (in encoding) weak-reduction-contrasimulation-impl-WOSou:}

\begin{itemize}
  \item fixes \(\text{Rel} ::= (**\text{procS}, **\text{procT}) \text{Proc} \times (**\text{procS}, **\text{procT}) \text{Proc} \) set
  \item and \(\text{TRel} ::= (**\text{procT} \times **\text{procT}) \) set
  \item assumes \(A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
    \(\land A2: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^{*}\)
    \(\land A3: \text{weak-reduction-contrasimulation } (\text{Rel}^{-1}) (\text{STCal Source Target})\)
  \item shows \(\text{weakly-operational-sound } (\text{TRel}^{*})\)

\end{itemize}

\textbf{proof clarify}

\begin{itemize}
  \item fix \(S T\)
  \item from \(A1\) have \(\text{(TargetTerm } ([S]), \text{SourceTerm } S) \in \text{Rel}^{-1}\)
    \(\text{by} \\textit{simp}\)

  \item moreover assume \([S] \rightarrow \text{Target} S T\)
  \item hence \(\text{TargetTerm } ([S]) \rightarrow (\text{STCal Source Target})^* (\text{TargetTerm } T)\)
    \(\text{by} \\textit{(simp add: STCal-steps(2))}\)

  \item ultimately obtain \(Q'\) where \(A5: \text{SourceTerm } S \rightarrow (\text{STCal Source Target})^{*} Q'\)
    \(\land A6: (Q', \text{TargetTerm } T) \in \text{Rel}^{-1}\)

  \item using \(A3\)
    \(\text{by} \\textit{blast}\)

  \item from \(A5\) obtain \(S'\) where \(A7: S' \in S Q'\) and \(A8: S \rightarrow \text{Source} S'\)
    \(\text{by} \\textit{(auto simp add: STCal-steps(1))}\)

  \item have \(Q' \rightarrow (\text{STCal Source Target})^{*} Q'\)
    \(\text{by} \\textit{(simp add: steps-refl)}\)

  \item with \(A6 \ A3\) obtain \(P''\) where \(A9: \text{TargetTerm } T \rightarrow (\text{STCal Source Target})^{*} P''\)
    \(\land A10: (P'', Q') \in \text{Rel}^{-1}\)
    \(\text{by} \\textit{blast}\)

  \item from \(A9\) obtain \(T'\) where \(A11: T' \in T P''\) and \(A12: T \rightarrow \text{Target} S T'\)
    \(\text{by} \\textit{(auto simp add: STCal-steps(2))}\)

  \item from \(A10\) have \((Q', P'') \in \text{Rel}\)
    \(\text{by} \\textit{induct}\)

  \item with \(A2 A7 A11\) have \([S'], \ T') \in \text{TRel}^{*}\)
    \(\text{by} \\textit{simp}\)

\end{itemize}
8.4 (Strong) Operational Soundness vs (Strong) Simulation

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

\textbf{lemma (in encoding) weak-reduction-simulation-impl-OSou:}

\begin{itemize}
  \item fixes \text{Rel} :: (\langle \text{proc}S, \text{proc}T \rangle \text{Proc} \times \langle \text{proc}S, \text{proc}T \rangle \text{Proc}) \text{ set}
  \item and \text{TRel} :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set}
  \item assumes \text{A1:} \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}
  \item and \text{A2:} \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*
  \item and \text{A3:} weak-reduction-simulation (\text{Rel}^{-1}) (\text{STCal Source Target})
  \item shows operational-sound (\text{TRel}^*)
\end{itemize}

\textbf{proof}

\begin{itemize}
  \item fix \text{S T}
  \item from \text{A1 have (TargetTerm } ([S]), \text{SourceTerm } S) \in \text{Rel}^{-1}
  \item by simp
  \item moreover assume \text{[S]} \rightarrow Target T
  \item hence TargetTerm ([S]) \rightarrow(\text{STCal Source Target}* (\text{TargetTerm } T)
  \item by (simp add: STCal-steps(2))
  \item ultimately obtain \text{Q'} where \text{A5: SourceTerm } S \rightarrow(\text{STCal Source Target}* \text{Q'}
  \item and \text{A6: (TargetTerm } T, \text{Q'}) \in \text{Rel}^{-1}
  \item using \text{A3}
  \item by blast
  \item from \text{A5 obtain \text{S'} where \text{A7: S'} S' and \text{A8: S' \rightarrow Source S'}}
  \item by (auto simp add: STCal-steps(1))
  \item from \text{A6 have (Q', TargetTerm } T) \in \text{Rel}
  \item by induct
  \item with \text{A2 A7 have ([S'], T) \in TRel*}
  \item by simp
  \item with \text{A8 show \exists S'. S \rightarrow Source S' \land ([S'], T) \in TRel*}
  \item by blast
\end{itemize}

\textbf{qed}

\textbf{lemma (in encoding) OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation:}

\begin{itemize}
  \item fixes \text{TRel} :: (\langle \text{proc}T \times \text{proc}T \rangle \text{ set}
  \item and \text{TRel}^* :: \text{weak-reduction-simulation ((TRel*)^{-1} Target)
  \item = weak-reduction-simulation ((\text{indRelRTPO TRel})^{-1}) (\text{STCal Source Target})
  \item shows operational-sound (\text{TRel}^*)
\end{itemize}

\textbf{proof (rule iffI, erule conjE)}

\begin{itemize}
  \item assume \text{os: operational-sound (TRel^*)}
  \item and \text{sim: weak-reduction-simulation ((TRel*)^{-1} Target}
  \item show weak-reduction-simulation ((\text{indRelRTPO TRel})^{-1}) (\text{STCal Source Target})
  \item proof clarify
  \item fix \text{P Q P'}
  \item assume \text{Q \leq [\cdot]} \text{RT}<\text{TRel} > P \text{ and P \rightarrow(\text{STCal Source Target}* P'}
  \item thus \text{\exists Q'. Q \rightarrow(\text{STCal Source Target}* Q' \land (P', Q') \in (\text{indRelRTPO TRel})^{-1}}
  \item proof (induct arbitrary: \text{P'})
  \item case (\text{encT S})
  \item assume TargetTerm ([S]) \rightarrow(\text{STCal Source Target}* \text{P'}
  \item from this obtain \text{T where A1: T \in T P' and A2: [S] \rightarrow Target T}
  \item by (auto simp add: STCal-steps(2))
  \item from \text{os A2 obtain S' where A3: S \rightarrow Source S' and A4: ([S'], T) \in TRel*}
  \item by blast
  \item from \text{A3 have SourceTerm S \rightarrow(\text{STCal Source Target}* (\text{SourceTerm S'})
  \item by (simp add: STCal-steps(1))
  \item moreover have SourceTerm S' \leq [\cdot] \text{RT}<\text{TRel} > P'}
\end{itemize}
proof
  from A4 have \([S'] = T \lor ([S'], T) \in T\rel^+\)
    using rtrancl-eq-or-trancl[of \([S']\ T\ T\rel]\)
    by blast
  moreover have \(A5: \text{SourceTerm } S' \leq [\cdot]RT < T\rel > \text{TargetTerm } ([S'])\)
    by (simp add: indRelRTPO.encR)
  with A1 have \([S'] = T \implies \text{SourceTerm } S' \leq [\cdot]RT < T\rel > P'\)
    by simp
  moreover have \(([S'], T) \in T\rel^+ \implies \text{SourceTerm } S' \leq [\cdot]RT < T\rel > P'\)
  proof
    assume \(([S'], T) \in T\rel^+\)
    hence TargetTerm \(([S']) \leq [\cdot]RT < T\rel > \text{TargetTerm } T\)
      by (rule transitive-closure-of-TRel-to-indRelRTPO)
    with A5 have \(\text{SourceTerm } S' \leq [\cdot]RT < T\rel > \text{TargetTerm } T\)
      by (rule indRelRTPO.trans)
    with A1 show \(\text{SourceTerm } S' \leq [\cdot]RT < T\rel > P'\)
      by simp
  qed
  ultimately show \(\text{SourceTerm } S' \leq [\cdot]RT < T\rel > P'\)
    by blast
  qed
  hence \((P', \text{SourceTerm } S') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by simp
  ultimately
  show \(\exists Q'. \text{SourceTerm } S \longmapsto (\text{STCal Source Target})^* Q' \land (P', Q') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by blast
next
case (source S)
  assume \(\text{SourceTerm } S \longmapsto (\text{STCal Source Target})^* P'\)
  moreover from this obtain \(S' \text{ where } B1: S' \in S P'\)
    by (auto simp add: STCal-steps(1))
  hence \((P', P') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by (simp add: indRelRTPO.source)
  ultimately
  show \(\exists Q'. \text{SourceTerm } S \longmapsto (\text{STCal Source Target})^* Q' \land (P', Q') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by blast
next
case (target \(T1 T2\))
  assume \(\text{TargetTerm } T2 \longmapsto (\text{STCal Source Target})^* P'\)
  from this obtain \(T2' \text{ where } C1: T2' \in T P' \land C2: T2 \longmapsto \text{Target}^* T2'\)
    by (auto simp add: STCal-steps(2))
  assume \((T1, T2) \in T\rel\)
  hence \((T2', T1) \in (T\rel^+)^{-1}\)
    by simp
  with \(C2 \text{ sim obtain } T1' \text{ where } C3: T1 \longmapsto \text{Target}^* T1' \land C4: (T2', T1') \in (T\rel^+)^{-1}\)
    by blast
  from C3 have \(\text{TargetTerm } T1 \longmapsto (\text{STCal Source Target})^* (\text{TargetTerm } T1')\)
    by (simp add: STCal-steps(2))
  moreover from C4 have \((T1', T2') \in T\rel^+\)
    by induct
  hence TargetTerm \(T1' \leq [\cdot]RT < T\rel > \text{TargetTerm } T2'\)
    by (rule transitive-closure-of-TRel-to-indRelRTPO)
  with C1 have \((P', \text{TargetTerm } T1') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by simp
  ultimately
  show \(\exists Q'. \text{TargetTerm } T1 \longmapsto (\text{STCal Source Target})^* Q' \land (P', Q') \in (\text{indRelRTPO} \ T\rel)^{-1}\)
    by blast
next
case (trans \(P Q R R')\)
  assume \(R \longmapsto (\text{STCal Source Target})^* R'\)
  and \(\exists R'. R \longmapsto (\text{STCal Source Target})^* R'\)
  169
\[ \exists Q', Q \mapsto (\text{STCal Source Target})^* \] $\land$ $(R', Q') \in (\text{indRelRTPO TRel})^{-1}$

From this obtain $Q'$ where $D1$: $Q \mapsto (\text{STCal Source Target})^*$ $\land$ $D2$: $(R', Q') \in (\text{indRelRTPO TRel})^{-1}$

By blast

Assume $\land Q', Q \mapsto (\text{STCal Source Target})^*$ $\implies \exists P'. P \mapsto (\text{STCal Source Target})^* P' \land (Q', P') \in (\text{indRelRTPO TRel})^{-1}$

With $D1$ obtain $P'$ where $D3$: $P \mapsto (\text{STCal Source Target})^* P'$ $\land$ $D4$: $(Q', P') \in (\text{indRelRTPO TRel})^{-1}$

By blast

Qed

Qed

Next

Have $\forall S. \text{SourceTerm } S \subseteq \llbracket RT < TRel \rrbracket \text{ TargetTerm } ([S])$

By (simp add: indRelRTPO.encR)

Moreover have $\forall S T. \text{SourceTerm } S \subseteq \llbracket RT < TRel \rrbracket \text{ TargetTerm } T \mapsto ([S], T) \in TRel^*$

Using indRelRTPO-to-TRel(2) [where TRel=TRel] trans-closure-of-TRel-refl-cond

By simp

Moreover

Assume sim: weak-reduction-simulation ((indRelRTPO TRel)^{-1}) (STCal Source Target)

Ultimately have operational-sound (TRel^*)

Using weak-reduction-simulation-impl-OSou [where Rel=indRelRTPO TRel and TRel=TRel]

By simp

Moreover from sim have weak-reduction-simulation ((TRel^+)^{-1}) Target

Using indRelRTPO-impl-TRel-is-weak-reduction-simulation-rev [where TRel=TRel]

By simp

Ultimately show operational-sound (TRel^*) $\land$ weak-reduction-simulation ((TRel^+)^{-1}) Target

By simp

Qed

Lemma (in encoding) OSou-iff-weak-reduction-simulation:

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

Shows (operational-sound (TRel^*) $\land$ weak-reduction-simulation ((TRel^+)^{-1}) Target)

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

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

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

$\land$ ($\forall S T. \text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \mapsto ([S], T) \in \text{TRel}^*$

$\land$ weak-reduction-simulation (Rel^{-1}) (STCal Source Target))

Proof (rule iff, erule conjE)

Have $\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelRTPO TRel}$

By (simp add: indRelRTPO.encR)

Moreover have $\forall T1 T2. (T1, T2) \in \text{TRel} \mapsto (\text{TargetTerm } T1 \subseteq \llbracket RT < TRel \rrbracket \text{ TargetTerm } T2$

By (simp add: indRelRTPO.target)

Moreover have $\forall T1 T2. \text{TargetTerm } T1 \subseteq \llbracket RT < TRel \rrbracket \text{ TargetTerm } T2 \mapsto (T1, T2) \in \text{TRel}^+$

Using indRelRTPO-to-TRel(4) [where TRel=TRel]

By simp

Moreover have $\forall S T. \text{SourceTerm } S \subseteq \llbracket RT < TRel \rrbracket \text{ TargetTerm } T \mapsto ([S], T) \in \text{TRel}^*$

Using indRelRTPO-to-TRel(2) [where TRel=TRel] trans-closure-of-TRel-refl-cond

By simp

Moreover assume operational-sound (TRel^*)

And weak-reduction-simulation ((TRel^+)^{-1}) Target

Hence weak-reduction-simulation ((indRelRTPO TRel)^{-1}) (STCal Source Target)

Using OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation [where TRel=TRel]

By simp

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

$\land$ ($\forall T1 T2. (T1, T2) \in \text{Rel} \mapsto (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}$

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

170
\begin{align*}
& \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^* \\
& \text{weak-reduction-simulation (Rel}^{-1}) (\text{STCal Source Target}) \\
& \text{by blast}
\end{align*}

\text{next}

\text{assume } \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})
\begin{align*}
& \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel}) \\
& \land (\forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+) \\
& \land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*) \\
& \text{weak-reduction-simulation (Rel}^{-1}) (\text{STCal Source Target})
\end{align*}

\text{from this obtain Rel where A1: } (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})
\begin{align*}
& \text{and A2: } (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel}) \\
& \text{and A3: } (\forall T_1 T_2. (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+) \\
& \text{and A4: } (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*)
\end{align*}

\text{and A5: weak-reduction-simulation (Rel}^{-1}) (\text{STCal Source Target})
\text{by blast}

\text{from A1 A4 A5 have operational-sound (TRel,+)}
\begin{align*}
& \text{using weak-reduction-simulation-impl-OSou[where Rel=Rel and TRel=TRel]} \\
& \text{by simp}
\end{align*}

\text{moreover from A2 A3 A5 have weak-reduction-simulation ((TRel,+)^{-1}) Target}
\begin{align*}
& \text{using rel-with-target-impl-transC-TRel-is-weak-reduction-simulation-rev}[\text{where Rel=Rel and TRel=TRel}] \\
& \text{by simp}
\end{align*}

\text{ultimately show operational-sound (TRel,+)} \land \text{weak-reduction-simulation ((TRel,+)^{-1}) Target}
\text{by simp}

\text{qed}

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

\text{lemma (in encoding) strong-reduction-simulation-impl-SOSou:}
\begin{align*}
\text{fixes Rel : } (\text{procS} \times \text{procT}) \text{ Proc} \times (\text{procS} \times \text{procT}) \text{ Proc} \text{ set} \\
& \text{and TRel : } (\text{procT} \times \text{procT}) \text{ set} \\
\text{assumes A1: } (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \\
& \text{and A2: } (\forall T_1 T_2. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \\
& \text{and A3: } (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*) \\
\text{shows strongly-operational-sound (TRel,+)}
\end{align*}

\text{proof clarify}
\text{fix S T}
\text{from A1 have (TargetTerm ([S]), SourceTerm S) \in Rel}^{-1}
\text{by simp}

\text{moreover assume } ([S] \rightarrow \text{Target T}) \\
\text{hence TargetTerm ([S]) \rightarrow (STCal Source Target) (TargetTerm T)} \\
\text{by (simp add: STCal-step(2))}

\text{ultimately obtain Q' where A5: SourceTerm S \rightarrow (STCal Source Target) Q'} \\
\text{and A6: (TargetTerm T, Q') \in Rel}^{-1}
\text{using A3}
\text{by blast}
\text{from A5 obtain S' where A7: S' \in S Q' and A8: S \rightarrow Source S'} \\
\text{by (auto simp add: STCal-step(1))}
\text{from A6 have (Q', TargetTerm T) \in Rel}
\text{by induct}
\text{with A2 A7 have ([S'], T) \in TRel}^* \\
\text{by simp}
\text{with A8 show } \exists S'. S \rightarrow Source S' \land ([S'], T) \in TRel}^* \\
\text{by blast}
\text{qed}

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

proof (rule iffI, erule conjE)
assume os: strongly-operational-sound (TRel*)
and sim: strong-reduction-simulation ((TRel')⁻¹) Target
show strong-reduction-simulation ((indRelRTPO TRel)⁻¹) (STCal Source Target)
proof clarify
fix P Q P'
major premise
moreover assume P ⇔ (STCal Source Target) P'
ultimately show ∃ Q'. Q ⇒ (STCal Source Target) Q' ∧ (P', Q') ∈ (indRelRTPO TRel)⁻¹
proof (induct arbitrary: P')
case (encR S)
assume TargetTerm ([S]) ⇒ (STCal Source Target) P'
from this obtain T where A1: T ∈ T P' and A2: [S] ⇒ Target T
by (auto simp add: STCal-step(2))
from os A2 obtain S' where A3: S ⇒ Source S' and A4: ([S'], T) ∈ TRel*
by blast
moreover have SourceTerm S' ⇒ (STCal Source Target) (SourceTerm S')
by (simp add: STCal-step(1))
moreover have SourceTerm S' ≤_{[T]< TRel} P'
moreover have [S'], T ∈ TRel⁺ ⇒ SourceTerm S' ≤_{[T]< TRel} P'
proof –
assume ([S'], T) ∈ TRel⁺
hence TargetTerm ([S']) ≤_{[T]< TRel} TargetTerm T
by (rule transitive-closure-of-TRel-to-indRelRTPO)
with A5 have SourceTerm S' ≤_{[T]< TRel} SourceTerm T
by (rule indRelRTPO.trans)
with A1 show SourceTerm S' ≤_{[T]< TRel} P'
by (simp add: STCal-step(1))
qed
ultimately show SourceTerm S' ≤_{[T]< TRel} P'
by blast
qed
hence (P', SourceTerm S') ∈ (indRelRTPO TRel)⁻¹
by simp
ultimately show ∃ Q'. SourceTerm S ⇒ (STCal Source Target) Q' ∧ (P', Q') ∈ (indRelRTPO TRel)⁻¹
by blast
next
case (source S)
assume SourceTerm S ⇒ (STCal Source Target) P'
major premise
moreover from this obtain S' where B1: S' ∈ S P'
by (auto simp add: STCal-step(1))
hence (P', P') ∈ (indRelRTPO TRel)⁻¹
by (simp add: indRelRTPO.source)
ultimately show ∃ Q'. SourceTerm S ⇒ (STCal Source Target) Q' ∧ (P', Q') ∈ (indRelRTPO TRel)⁻¹
by blast
next
case (target T1 T2)
assume TargetTerm T2 ⇒ (STCal Source Target) P'
from this obtain $T_2'$ where $C_1$: $T_2' \in T \ P'$ and $C_2$: $T_2 \rightarrow \text{Target} \ T_2'$
by (auto simp add: STCal-step(2))
assume $(T_1, T_2) \in \text{TRel}$
hence $(T_2', T_1) \in (\text{TRel}^+)^{-1}$
by simp
with $C_2$ sim obtain $T_1'$ where $C_3$: $T_1 \rightarrow \text{Target} \ T_1'$ and $C_4$: $(T_2', T_1') \in (\text{TRel}^+)^{-1}$
by blast
from $C_3$ have TargetTerm $T_1 \rightarrow$ (STCal Source Target) (TargetTerm $T_1'$)
by (simp add: STCal-step(2))
moreover from $C_4$ have $(T_1', T_2') \in \text{TRel}^+$
by induct
hence TargetTerm $T_1' \leq\{\VirSort RT<\text{TRel}\} \text{ TargetTerm} \ T_2'$
by (rule transitive-closure-of-\text{TRel-to-indRelRTPO})
with $C_1$ have $(P', \text{TargetTerm} \ T_1') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
by simp
ultimately
show $\exists Q'. \ \text{TargetTerm} \ T_1 \rightarrow$ (STCal Source Target) $Q' \land (P', Q') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
by blast
next
case (trans $P \ Q \ R \ R'$)
assume $R \rightarrow$ (STCal Source Target) $R'$
and $\langle R', R \rightarrow$ (STCal Source Target) $R' \rangle$

implies $\exists Q'. \ Q \rightarrow$ (STCal Source Target) $Q' \land (R', Q') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
from this obtain $Q'$ where $D_1$: $Q \rightarrow$ (STCal Source Target) $Q'$
and $D_2$: $(R', Q') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
by blast
assume $\langle Q', Q \rangle \rightarrow$ (STCal Source Target) $Q'$
implies $\exists P'. \ P \rightarrow$ (STCal Source Target) $P' \land (Q', P') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
with $D_1$ obtain $P'$ where $D_3$: $P \rightarrow$ (STCal Source Target) $P'$
and $D_4$: $(Q', P') \in (\text{indRelRTPO} \ \text{TRel})^{-1}$
by blast
qed
qed
next
have $\forall S. \ \text{SourceTerm} \ S \leq\{\VirSort RT<\text{TRel}\} \text{ TargetTerm} \ ([S])$
by (simp add: indRelRTPO.encR)
moreover have $\forall S \ T. \ \text{SourceTerm} \ S \leq\{\VirSort RT<\text{TRel}\} \text{ TargetTerm} \ T \rightarrow ([S], T) \in \text{TRel}^*$
using indRelRTPO-to-TRel(2)[where $\text{Rel}=\text{TRel}$] trans-closure-of-\text{TRel-refl-cond}
by simp
moreover
assume sim: strong-reduction-simulation ((indRelRTPO TRel)$^{-1}$) (STCal Source Target)
ultimately have strongly-operational-sound ($\text{TRel}^*$)
using strong-reduction-simulation-impl-SOSou[where $\text{Rel}=\text{indRelRTPO} \ \text{TRel}$ and $\text{Rel}=\text{TRel}$]
by simp
moreover from sim have strong-reduction-simulation ((($\text{TRel}^*$)$^{-1}$) Target
using indRelRTPO-impl-TRel-is-strong-reduction-simulation-rev[where $\text{Rel}=\text{TRel}$]
by simp
ultimately
show strongly-operational-sound ($\text{TRel}^*$) \land strong-reduction-simulation ((($\text{TRel}^*$)$^{-1}$) Target
by simp
qed

lemma (in encoding) SOSou-iff-strong-reduction-simulation:
  fixes $\text{TRel} :: ('\text{proc}T \times '\text{proc}T) \text{ set}$
  shows (strongly-operational-sound ($\text{TRel}^*$) \land strong-reduction-simulation ((($\text{TRel}^*$)$^{-1}$) Target)
  $= (\exists \text{Rel}. \ (\forall S. \ (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S]) \in \text{Rel})$
  \land ($\forall T_1 T_2. \ (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm} \ T_1, \text{TargetTerm} \ T_2) \in \text{Rel}$)

173
\(\wedge (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in T\text{Rel}^+)\)
\(\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T\text{Rel}^+)\)
\(\wedge \text{strong-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})\)

**proof** (rule iff, erule conjE)

**have** \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelRTPO } T\text{Rel}\)
**by** (simp add: \text{indRelRTPO.encR})

**moreover have** \(\forall T_1 T_2. (T_1, T_2) \in T\text{Rel} \rightarrow \text{TargetTerm } T_1 \preceq [\cdot]RT\preceq T\text{Rel} > \text{TargetTerm } T_2\)
**by** (simp add: \text{indRelRTPO.target})

**moreover have** \(\forall T_1 T_2. \text{TargetTerm } T_1 \preceq [\cdot]RT\preceq T\text{Rel} > \text{TargetTerm } T_2 \rightarrow (T_1, T_2) \in T\text{Rel}^+\)
**using** \text{indRelRTPO-to-TRel4}[\text{where } T\text{Rel}=T\text{Rel}]\)
**by** simp

**moreover assume** \text{strongly-operational-sound } (T\text{Rel}^*)
\(\wedge \text{strong-reduction-simulation } ((T\text{Rel}^+)^{-1}) \text{Target}\)

**hence** \text{strong-reduction-simulation } ((\text{indRelRTPO } T\text{Rel})^{-1}) \text{(STCal Source Target)}
**using** \text{SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation}[\text{where } T\text{Rel}=T\text{Rel}]\)
**by** simp

**ultimately show** \(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \land (\forall T_1 T_2. (T_1, T_2) \in \text{Rel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \land (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in T\text{Rel}^+) \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T\text{Rel}^+) \wedge \text{strong-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})\)
**by** blast

**next**
**assume** \(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \land (\forall T_1 T_2. (T_1, T_2) \in \text{Rel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \land (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in T\text{Rel}^+) \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T\text{Rel}^+) \wedge \text{strong-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})\)

**from this obtain** \text{Rel when} \text{A1: } \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}
\text{ and } \text{A2: } \forall T_1 T_2. (T_1, T_2) \in T\text{Rel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}
\text{ and } \text{A3: } \forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in T\text{Rel}^+
\text{ and } \text{A4: } \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T\text{Rel}^+
\text{ and } \text{A5: } \text{strong-reduction-simulation } (\text{Rel}^{-1}) (\text{STCal Source Target})\)
**by** blast

**from A1 A4 A5 have** \text{strongly-operational-sound } (T\text{Rel}^*)
**using** \text{strong-reduction-simulation-impl-SOSou}[\text{where } \text{Rel}=\text{Rel} \land \text{TRel}=\text{TRel}]\)
**by** simp

**moreover from** \text{A2 A3 A5 have} \text{strong-reduction-simulation } ((T\text{Rel}^+)^{-1}) \text{Target}
**using** \text{rel-with-target-impl-transC-TRel-is-strong-reduction-simulation-rev}[\text{where } \text{Rel}=\text{Rel} \land \text{TRel}=\text{TRel}]\)
**by** simp

**ultimately**
**show** \text{strongly-operational-sound } (T\text{Rel}^*) \land \text{strong-reduction-simulation } ((T\text{Rel}^+)^{-1}) \text{Target}
**by** simp

**qed**

**lemma** (in encoding) \text{SOSou-modulo-TRel-iff-strong-reduction-simulation:}
**shows** \(\exists \text{TRel. strongly-operational-sound } (\text{TRel}^*) \land \text{strong-reduction-simulation } ((\text{TRel}^+)^{-1}) \text{Target}\)
**proof** (rule iff)
**assume** \(\exists \text{TRel. strongly-operational-sound } (\text{TRel}^*) \land \text{strong-reduction-simulation } ((\text{TRel}^+)^{-1}) \text{Target}\)
**from this obtain** \text{TRel when} \text{strongly-operational-sound } (\text{TRel}^*) \land \text{strong-reduction-simulation } ((\text{TRel}^+)^{-1}) \text{Target}
**by** blast
hence strong-reduction-simulation ((indRelRTPO TRel)\(^{-1}\)) (STCal Source Target)
using SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation[where TRel=TRel]
by simp
moreover have \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelRTPO TRel}\)
by (simp add: indRelRTPO_encR)
moreover have \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO TRel}\)
\(\longrightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in (\text{indRelRTPO TRel})\)
using indRelRTPO-relates-source-target[where TRel=TRel]
by simp
ultimately show \(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\)
\& \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}\)
\(\longrightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}^\ast\)
\& strong-reduction-simulation (Rel\(^{-1}\)) (STCal Source Target)
by blast

next
assume \(\exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\)
\& \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}\)
\(\longrightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}^\ast\)
\& strong-reduction-simulation (Rel\(^{-1}\)) (STCal Source Target)
from this obtain Rel where A1: \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
and A2: \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel}\)
\(\longrightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in \text{Rel}^\ast\)
and A3: strong-reduction-simulation (Rel\(^{-1}\)) (STCal Source Target)
by blast
from A2 obtain TRel where \(\forall T1 T2. (T1, T2) \in \text{TRel} \longrightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\)
and \(\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \longrightarrow (T1, T2) \in \text{TRel}^\ast\)
and \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
using target-relation-from-source-target-relation[where Rel=Rel]
by blast
with A1 A3
have strongly-operational-sound (TRel\(^\ast\)) \& strong-reduction-simulation ((TRel\(^\ast\))\(^{-1}\)) Target
using SOSou-iff-strong-reduction-simulation[where TRel=TRel]
by blast
thus \(\exists \text{TRel}. \text{strongly-operational-sound (TRel\(^\ast\)) \& strong-reduction-simulation ((TRel\(^\ast\))\(^{-1}\)) Target}\)
by blast
qed

8.5 Weak Operational Correspondence vs Correspondence Similarity

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

lemma (in encoding) weak-red-correspondence-simulation-impl-WOC:
fixes Rel :: ('procS', 'procT) Proc \times ('procS', 'procT) Proc set
and TRel :: ('procT \times 'procT) set
assumes enc: \(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}\)
and rRel: \(\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
and es: weak-reduction-correspondence-simulation Rel (STCal Source Target)
shows weakly-operational-corresponding (TRel\(^\ast\))
proof
from enc rRel es show operational-complete (TRel\(^\ast\))
using weak-red-simulation-impl-OCom[where TRel=TRel]
by simp
next
show weakly-operational-sound (TRel\(^\ast\))
proof clarify
fix S T
from enc have (SourceTerm S, TargetTerm ([S])) \in Rel
by simp
moreover assume [S] \longrightarrow Target\(^\ast\) T

qed
hence TargetTerm ([S]) \rightarrow (STCal \ Source \ Target)* (TargetTerm T)
by (simp add: STCal-steps(2))
ultimately obtain P' Q' where A1: SourceTerm S \rightarrow (STCal \ Source \ Target)* P'
and A2: TargetTerm T \rightarrow (STCal \ Source \ Target)* Q' and A3: (P', Q') \in Rel
using cs
by blast
from A1 obtain S' where A4: S' \in S P' and A5: S \rightarrow Source* S'
by (auto simp add: STCal-steps(1))
from A2 obtain T' where A6: T' \in T Q' and A7: T \rightarrow Target* T'
by (auto simp: STCal-steps(2))
from tRel A3 A4 A6 have ([S'], T') \in TRel*
by simp
with A5 A7 show \exists S' T'. S \rightarrow Source* S' \land T \rightarrow Target* T' \land ([S'], T') \in TRel*
by blast
qed

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

lemma (in encoding) WOC-iff-indRelRTPO-is-reduction-correspondence-simulation:
fixes TRel :: ('procT \times 'procT) set
shows (weakly-operational-corresponding (TRel*)
\land weak-reduction-correspondence-simulation (TRel+) Target)
= weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)
proof (rule iffI, erule conjE)
assume voc: weakly-operational-corresponding (TRel*)
and csi: weak-reduction-correspondence-simulation (TRel+) Target
show weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)
proof from voc csi show sim: weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
using OCom-iff-indRTPO-is-weak-reduction-simulation[where TRel=TRel]
by simp
show \forall P Q Q'. P \leq [\overline{\bot}]RT<TRel> Q \land Q \rightarrow (STCal Source Target)* Q'
\rightarrow (\exists P'' Q''). P \rightarrow (STCal Source Target)* P'' \land Q' \rightarrow (STCal Source Target)* Q''
\land P'' \leq [\overline{\bot}]RT<TRel> Q''
proof clarify
fix P Q Q'
assumption P \leq [\overline{\bot}]RT<TRel> Q and Q \rightarrow (STCal Source Target)* Q'
thus \exists P'' Q''. P \rightarrow (STCal Source Target)* P'' \land Q' \rightarrow (STCal Source Target)* Q''
\land P'' \leq [\overline{\bot}]RT<TRel> Q''
proof (induct arbitrary: Q')
case (eneR S)
assume TargetTerm ([S]) \rightarrow (STCal Source Target)* Q'
from this obtain T' where A1: T \in T Q' and A2: [S] \rightarrow Target* T'
by (auto simp add: STCal-steps(2))
from A2 voc obtain S' T' where A3: S \rightarrow Source* S' and A4: T \rightarrow Target* T'
and A5: ([S'], T') \in TRel*
by blast
from A3 have SourceTerm S \rightarrow (STCal Source Target)* (SourceTerm S')
by (simp add: STCal-steps(1))
moreover from A4 have TargetTerm T \rightarrow (STCal Source Target)* (TargetTerm T')
by (simp add: STCal-steps(2))
moreover have SourceTerm S' \leq [\overline{\bot}]RT<TRel> TargetTerm T'
proof have A6: SourceTerm S' \leq [\overline{\bot}]RT<TRel> TargetTerm ([S'])
by (rule indRelRTPO.eneR)
from A5 have [S'] = T' \lor ([S'], T') \in TRel+
using rtrancl-eq-or-rtrancl[of [S'] T' TRel]
by blast

176
moreover from A6 have $[S'] = T' \Rightarrow \text{SourceTerm } S' \lesssim \cdot \text{TargetTerm } T'$
by simp
moreover have $(T', T') \in TRel^+ \Rightarrow \text{SourceTerm } S' \lesssim \cdot \text{TargetTerm } T'$
proof –
  assume $(T', T') \in TRel^+$
  hence $\text{TargetTerm } (T') \lesssim \cdot \text{TargetTerm } T'$
  by (simp add: transitive-closure-of-TRel-to-indRelRTPO[where TRel=TRel])
with A6 show $\text{SourceTerm } S' \lesssim \cdot \text{TargetTerm } T'$
  by (rule indRelRTPO.trans)
qed
ultimately show $\exists P'' \ Q''. \text{SourceTerm } S \longrightarrow (\text{STCal Source Target})* P''$
  \land \ Q'' \longrightarrow (\text{STCal Source Target})* Q'' \land P'' \lesssim \cdot \text{TargetTerm } Q''$
  using A1
  by blast
next
case (source S)
  assume B1: $\text{SourceTerm } S \longrightarrow (\text{STCal Source Target})* Q'$
moreover have $Q' \longrightarrow (\text{STCal Source Target})* Q'$
  by (rule steps-refl)
moreover from B1 obtain $S'$ where $S' \in S Q'$
  by (auto simp add: STCal-steps(1))
  hence $Q' \lesssim \cdot \text{TargetTerm } Q''$
  by (simp add: indRelRTPO.source)
ultimately show $\exists P'' \ Q''. \text{SourceTerm } S \longrightarrow (\text{STCal Source Target})* P''$
  \land \ Q'' \longrightarrow (\text{STCal Source Target})* Q'' \land P'' \lesssim \cdot \text{TargetTerm } Q''$
  by blast
next
case (target T1 T2)
  assume TargetTerm T2 \longrightarrow (\text{STCal Source Target})* Q'
from this obtain T2' where C1: $T2' \in T Q'$ and C2: $T2 \longrightarrow \text{TargetTerm } T2'$
  by (auto simp add: STCal-steps(2))
assume $(T1, T2') \in TRel$
  hence $(T1, T2') \in TRel^+$
  by simp
with C2 csi obtain $T1' \ T2''$ where C3: $T1 \longrightarrow \text{TargetTerm } T1'$ and C4: $T2' \longrightarrow \text{TargetTerm } T2''$
  and C5: $(T1', T2'') \in TRel^+$
  by blast
from C3 have $\text{TargetTerm } T1 \longrightarrow (\text{STCal Source Target})* (\text{TargetTerm } T1')$
  by (simp add: STCal-steps(2))
moreover from C1 C4 have $Q' \longrightarrow (\text{STCal Source Target})* (\text{TargetTerm } T2'')$
  by (simp add: STCal-steps(2))
moreover from C5 have $\text{TargetTerm } T1' \lesssim \cdot \text{TargetTerm } T2''$
  by (simp add: transitive-closure-of-TRel-to-indRelRTPO)
ultimately show $\exists P'' \ Q''. \text{SourceTerm } T1 \longrightarrow (\text{STCal Source Target})* P''$
  \land \ Q'' \longrightarrow (\text{STCal Source Target})* Q'' \land P'' \lesssim \cdot \text{TargetTerm } Q''$
  by blast
next
case (trans P Q R R')
  assume R \longrightarrow (\text{STCal Source Target})* R'
  \land \ R': R \longrightarrow (\text{STCal Source Target})* R' \Rightarrow \exists Q'' \ R''. Q \longrightarrow (\text{STCal Source Target})* Q''
  \land \ R' : (\text{STCal Source Target})* R'' \land Q'' \lesssim \cdot \text{TargetTerm } R''$
  and \ Q': Q \longrightarrow (\text{STCal Source Target})* Q' \Rightarrow \exists P'' \ Q''. P \longrightarrow (\text{STCal Source Target})* P''
  \land \ Q' \longrightarrow (\text{STCal Source Target})* Q'' \land P'' \lesssim \cdot \text{TargetTerm } Q''$
moreover have trans (indRelRTPO TRel)
  using indRelRTPO.trans
  unfolding trans-def
  by blast
ultimately show ?case
using sim reduction-correspondence-simulation-condition-trans[where P=P and
Rel=indRelRTPO TRel and Cal=STCal Source Target and Q=Q and R=R]

by blast
qed
qed

next

assume csi: weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)

show weakly-operational-corresponding (TRel⁺)
∧ weak-reduction-correspondence-simulation (TRel⁺) Target

proof

have ∀ S. SourceTerm S ≤∥_∥<TRel> TargetTerm ([S])
by (simp add: indRelRTPO.encR)

moreover have ∀ T1 T2. (T1, T2) ∈ TRel —>(TargetTerm T1, TargetTerm T2) ∈ Rel
by (simp add: indRelRTPO.target)

moreover have ∀ T1 T2. TargetTerm T1 ≤∥_∥<TRel> TargetTerm T2 —>(T1, T2) ∈ TRel⁺
using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond
by simp

ultimately show weakly-operational-corresponding (TRel⁺)
using weak-reduction-correspondence-simulation-iml-WOC[where Rel=indRelRTPO TRel and
TRel=TRel] csi
by simp

next

from csi show weak-reduction-correspondence-simulation (TRel⁺) Target
using indRelRTPO-impl-TRel-is-weak-reduction-correspondence-simulation[where TRel=TRel]
by simp
qed

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

fixes TRel :: ('procT × 'procT) set

shows (weakly-operational-corresponding (TRel⁺))
∧ weak-reduction-correspondence-simulation (TRel⁺) Target
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —>(TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —>(T1, T2) ∈ TRel⁺)
∧ (∃ S T. (SourceTerm S, TargetTerm T) ∈ Rel —>([S], T) ∈ TRel⁺)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)

proof (rule iffI,erule conjE)

have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ indRelRTPO TRel
by (simp add: indRelRTPO.encR)

moreover have ∀ T1 T2. (T1, T2) ∈ TRel —>(TargetTerm T1, TargetTerm T2) ∈ Rel
by (simp add: indRelRTPO.target)

moreover have ∀ T1 T2. TargetTerm T1 ≤∥_∥<TRel> TargetTerm T2 —>(T1, T2) ∈ TRel⁺
using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond
by simp

moreover assume weakly-operational-corresponding (TRel⁺)
and weak-reduction-correspondence-simulation (TRel⁺) Target

hence weak-reduction-correspondence-simulation (indRelRTPO TRel) (STCal Source Target)
using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where TRel=TRel]
by simp

ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —>(TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —>(T1, T2) ∈ TRel⁺)
∧ (∃ S T. (SourceTerm S, TargetTerm T) ∈ Rel —>([S], T) ∈ TRel⁺)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)
by blast

next

assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —>(TargetTerm T1, TargetTerm T2) ∈ Rel)
\(\forall T1. T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+\)
\(\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+\)

From this obtain \(\text{Rel} \iff A1: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}\)
and \(A2: \forall T1. T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\)
and \(A3: \forall T1. T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+\)
and \(A4: \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+\)
and \(A5: \text{weak-reduction-correspondence-simulation} \text{Rel} (\text{STCal Source Target})\)

by \text{blast}

from \(A1 A4 A5\) have weakly-operational-corresponding \((\text{TRel}^+)\)
using weak-reduction-correspondence-simulation-impl-WOC[where \(\text{Rel} = \text{Rel}\) and \(\text{TRel} = \text{TRel}\)]
by \text{simp}

moreover from \(A2 A3 A5\) have weak-reduction-correspondence-simulation \((\text{TRel}^+)\) Target
using rel-with-target-impl-transC-\text{TRel-is-weak-reduction-correspondence-simulation}
by \text{simp}

ultimately show weakly-operational-corresponding \((\text{TRel}^+)\) Target

by \text{simp}

qed

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

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

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

\text{assumes} \text{transT}: \text{trans} \text{TRel}

\text{shows} ((\forall T1. T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel})
\land (\forall T1. T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+))
\Rightarrow (\text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\})

\text{proof (rule iffI, erule conjE)}

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

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

\text{with} \text{transT} \text{show} \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}

using trancl-id[of \text{TRel}]

by \text{blast}

next

\text{assume} \(A1\): \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}

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

by \text{simp}

moreover from \text{transT} \(A\)

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

using trancl-id[of \text{TRel}]

by \text{blast}

ultimately show \(\forall T1. T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\)

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

by \text{simp}

qed

lemma \text{(in encoding) WOC-wrt-preorder-iff-reduction-correspondence-simulation:}

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

\text{shows} (weakly-operational-corresponding \text{TRel} \land \text{preorder} \text{TRel}
\land weak-reduction-correspondence-simulation \text{TRel} Target)
\Rightarrow (\exists \text{Rel}. (\forall S. \text{operational-complete} \text{TRel} \land \text{preorder} \text{TRel}
\land weak-reduction-correspondence-simulation \text{TRel} Target)
\land \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
\land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})
\land \text{preorder} \text{Rel}
\land weak-reduction-correspondence-simulation \text{Rel} (\text{STCal Source Target}))

\text{proof (rule iffI, erule conjE, erule conjE, erule conjE)}

\text{assume} \(A1\): \text{operational-complete} \text{TRel} \land \text{A2: weakly-operational-sound} \text{TRel}

\text{and} \text{\text{A3: preorder} \text{TRel} \land \text{A4: weak-reduction-correspondence-simulation} \text{TRel} Target}

from \text{A3} have \(A5\): \text{TRel}^+ = \text{TRel}

using trancl-id[of \text{TRel}]

unfolding preorder-on-def

179
by blast

with $A3$ have $TRel^+ = TRel$
  using trancl-id[of $TRel$] reflcl-trancl[of $TRel$]
  unfolding preorder-on-def refl-on-def
  by auto

with $A1$ $A2$ have weakly-operational-corresponding ($TRel^+$)
  by simp

moreover from $A4$ $A5$ have weak-reduction-correspondence-simulation ($TRel^+$) Target
  by simp

ultimately have weak-reduction-correspondence-simulation (indRelRTPO $TRel$) (STCal Source Target)
  using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where $TRel = TRel$]
  by blast

moreover have $\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelRTPO } TRel$
  by (simp add: indRelRTPO.encR)

moreover have $TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{indRelRTPO } TRel\}$

proof auto

fix $TP$ $TQ$

assume $(TP, TQ) \in TRel$

thus $\text{TargetTerm } TP \leq_T\text{TRel} < TRel > \text{TargetTerm } TQ$
  by (rule indRelRTPO.target)

next

fix $TP$ $TQ$

assume $\text{TargetTerm } TP \leq_T\text{TRel} < TRel > \text{TargetTerm } TQ$

with $A3$ show $(TP, TQ) \in TRel$
  using indRelRTPO-to-TRel(4)[where $TRel = TRel$] trancl-id[of $TRel$]
  unfolding preorder-on-def refl-on-def
  by blast

qed

moreover from $A3$

have $\forall S \text{ T}. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO } TRel \rightarrow ([S], T) \in TRel^+$
  using indRelRTPO-to-TRel(2)[where $TRel = TRel$] reflcl-trancl[of $TRel$]
  unfolding preorder-on-def refl-on-def
  by blast

with $A3$ have $\forall S \text{ T}. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{indRelRTPO } TRel \rightarrow ([S], T) \in TRel$
  using trancl-id[of $TRel$]
  unfolding preorder-on-def
  by blast

moreover from $A3$ have refl (indRelRTPO $TRel$)
  using indRelRTPO-refl[of $TRel$]
  unfolding preorder-on-def
  by simp

moreover have trans (indRelRTPO $TRel$)
  using indRelRTPO.trans
  unfolding trans-def
  by blast

ultimately show $\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S]))) \in \text{Rel})$
  $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$
  $\wedge (\forall S \text{ T}. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in TRel)$
  $\wedge \text{preorder Rel}$
  $\wedge \text{weak-reduction-correspondence-simulation Rel } (\text{STCal Source Target})$
  unfolding preorder-on-def
  by blast

next

assume $\exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S]))) \in \text{Rel})$
  $\wedge TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$
  $\wedge (\forall S \text{ T}. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in TRel)$
  $\wedge \text{preorder Rel}$
  $\wedge \text{weak-reduction-correspondence-simulation Rel } (\text{STCal Source Target})$
from this obtain Rel where B1: \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \)
and B2: \( \text{TRel} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\} \)
and B3: \( \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel} \) and B4: preorder Rel
and B5: weak-reduction-correspondence-simulation Rel (STCal Source Target)

by blast

from B2 B4 have B6: refl TRel
  unfolding preorder-on-def refl-on-def
  by blast

from B2 B4 have B7: trans TRel
  unfolding trans-def preorder-on-def
  by blast

hence B8: \( TRel^+ = TRel \)
  using trancl-id[of TRel]
  by simp

with B6 have TRel^* = TRel
  unfolding reflcl-trancl[of TRel]
  by blast

with B1 B3 B5 have weakly-operational-corresponding TRel
  using weak-reduction-correspondence-simulation-impl-WOC[where Rel=Rel and TRel=TRel]
  by simp

moreover from B6 B7 have preorder TRel
  unfolding preorder-on-def
  by blast

moreover from B2 B5 B7 B8 have weak-reduction-correspondence-simulation TRel Target
  using rel-includes-TRel-modulo-preorder[where Rel=Rel and TRel=TRel]
  rel-with-target-impl-transC-TRel-is-weak-reduction-correspondence-simulation[where Rel=Rel and TRel=TRel]
  by fast

ultimately show weakly-operational-corresponding TRel \( \land \) preorder TRel
  \( \land \) weak-reduction-correspondence-simulation TRel Target
  by blast

qed

8.6 (Strong) Operational Correspondence vs (Strong) Bisimilarity

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

lemma (in encoding) OC-iff-indRelRTPO-is-weak-reduction-bisimulation:
  fixes TRel :: \((\text{procT} \times \text{procT})\) set
  shows (operational-corresponding \( (TRel)^* \)
  \( \land \) weak-reduction-bisimulation \( (TRel^+)^{-1} \) Target)
  = weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)

proof (rule iffI, erule conjE)
  assume occrr: operational-corresponding \( (TRel)^* \)
  and bisim: weak-reduction-bisimulation \( (TRel^+)^{-1} \) Target
  hence weak-reduction-simulation (indRelRTPO TRel) (STCal Source Target)
    using OCom-iff-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]
    by simp

moreover from bisim have weak-reduction-simulation \( ((TRel^+)^{-1}) \) Target
  using weak-reduction-bisimulations-impl-inverse-is-simulation[where Rel=TRel^+]
  by simp

with occrr have weak-reduction-simulation \( ((\text{indRelRTPO TRel})^{-1}) \) (STCal Source Target)
  using OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]
  by simp

ultimately show weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
  using weak-reduction-simulations-impl-bisimulation[where Rel=indRelRTPO TRel]
  by simp
next
assume bisim: weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
hence operational-complete (TRel*) ∧ weak-reduction-simulation (TRel+) Target
  using OCom-iff-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]
  by simp
moreover have weak-reduction-simulation ((indRelRTPO TRel)^−1) (STCal Source Target)
  using weak-reduction-bisimulations-impl-inverse-is-simulation[where Rel=indRelRTPO TRel]
  by simp
hence operational-sound (TRel*) ∧ weak-reduction-simulation ((TRel^−1)^−1) Target
  using OSou-iff-inverse-of-indRelRTPO-is-weak-reduction-simulation[where TRel=TRel]
  by simp
ultimately show operational-corresponding (TRel*) ∧ weak-reduction-bisimulation (TRel+) Target
  using weak-reduction-simulations-impl-bisimulation[where Rel=TRel^+]
  by simp
qed

lemma (in encoding) OC-iff-weak-reduction-bisimulation:
  fixes TRel :: (‘procT × ‘procT) set
  shows (operational-corresponding (TRel^*) ∧ weak-reduction-bisimulation (TRel^+) Target)
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∃ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^+)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target))
proof (rule iffI, erule conjE)
  have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ indRelRTPO TRel
    by (simp add: indRelRTPO.encR)
  moreover have ∀ T1 T2. (T1, T2) ∈ TRel → TargetTerm T1 ≲{1}RT<TRel> TargetTerm T2
    by (simp add: indRelRTPO.target)
  moreover have ∀ T1 T2. TargetTerm T1 ≲{1}RT<TRel> TargetTerm T2 → (T1, T2) ∈ TRel^+
    using indRelRTPO-to-TRel(4)[where TRel=TRel]
    by simp
  moreover have ∀ S T. SourceTerm S ≲{1}RT<TRel> TargetTerm T → ([S], T) ∈ TRel^*
    using indRelRTPO-to-TRel(2)[where TRel=TRel] trans-closure-of-TRel-refl-cond
    by simp
  moreover assume operational-corresponding (TRel^*)
  and weak-reduction-bisimulation (TRel^+) Target
  hence weak-reduction-bisimulation (indRelRTPO TRel) (STCal Source Target)
    using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel=TRel]
    by simp
ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^*)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
    by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^*)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
from this obtain Rel where A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and A2: ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
and A3: ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel^+
and A4: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^*
and A5: weak-reduction-bisimulation Rel (STCal Source Target)
  by blast
hence operational-complete (TRel^*)
  ∧ weak-reduction-simulation (TRel^+) Target
using $OCom$-iff-weakening-simulation [where $TRel = TRel$]
by blast

moreover from A5 have weak-reduction-simulation $(RRel^\leftarrow)$ $(STCal Source Target)
using weak-reduction-bisimulations-impl-inverse-is-simulation [where $Rel = TRel$]
by simp

with A1 A2 A3 A4 have operational-sound $(TRel^+)$
  $\land$ weak-reduction-simulation $((TRel^+)\leftarrow)$ $Target$
using $OSou$-iff-weakening-simulation [where $TRel = TRel$]
by blast

ultimately show operational-corresponding $(TRel^+)$
  $\land$ weak-reduction-bisimulation $(TRel^+) Target$
using weak-reduction-simulations-impl-bisimulation [where $Rel = TRel^+$]
by simp

qed

lemma (in encoding) OC-wrt-preorder-iff-weakening-simulation:
fixes $TRel :: (’procT \times ’procT) set$
shows (operational-corresponding $TRel \land$ preorder $TRel$
  $\land$ weak-reduction-bisimulation $TRel Target$
  $= (\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel$
  $\land$ $TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel\}$
  $\land$ $\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel$
  $\land$ preorder $Rel$
  $\land$ weak-reduction-bisimulation $Rel (STCal Source Target)$)
proof (rule iffI, erule conjE, erule conjE, erule conjE)
assume A1: operational-complete $TRel$ and A2: operational-sound $TRel$
and A3: preorder $TRel$ and A4: weak-reduction-bisimulation $TRel Target$
from A3 have A5: $TRel^+ = TRel$
  using trancl-id[of $TRel$]
  unfolding preorder-on-def
  by blast

with A3 have $TRel^+ = TRel$
  using refl-trancl[of $TRel$]
  unfolding preorder-on-def refl-on-def
  by blast

with A1 A2 have operational-corresponding $(TRel^+)$
  by simp

moreover from A4 A5 have weak-reduction-bisimulation $(TRel^+)$ Target
  by simp

ultimately have weak-reduction-bisimulation (indRelRTPO $TRel$) (STCal Source Target)
  using OC-iff-indRelRTPO-is-weakening-bisimulation [where $TRel = TRel$]
  by blast

moreover have $\forall S. SourceTerm S \sqsubseteq [\cdot]RT < TRel > TargetTerm ([S])$
  by (simp add: indRelRTPO_encR)

moreover have $TRel = \{(T1, T2). TargetTerm T1 \sqsubseteq [\cdot]RT < TRel > TargetTerm T2\}$
proof auto
  fix TP TQ
  assume (TP, TQ) $\in$ $TRel$
  thus $TargetTerm TP \sqsubseteq [\cdot]RT < TRel > TargetTerm TQ$
    by (rule indRelRTPO_target)
next
  fix TP TQ
  assume $TargetTerm TP \sqsubseteq [\cdot]RT < TRel > TargetTerm TQ$
  with A3 show (TP, TQ) $\in$ $TRel$
    using indRelRTPO-to-$TRel$(4) [where $TRel = TRel$] trancl-id[of $TRel$]
    unfolding preorder-on-def
    by blast

qed

moreover from A3
∀ S T. SourceTerm S ⊣_RT<TRel> TargetTerm T → ([S], T) ∈ TRel^+
  using indRelRTPO-to-TRel C TRel=TRel refl-trancl[of TRel]
  trans-closure-of-TRel-refl-cond [where TRel=TRel]
  unfolding preorder-on-def refl-on-def
  by auto
with A3 have ∀ S T. SourceTerm S ⊣_RT<TRel> TargetTerm T → ([S], T) ∈ TRel
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
moreover have refl (indRelRTPO TRel)
  unfolding preorder-on-def
  by (simp add: indRelRTPO-refl)
moreover have trans (indRelRTPO TRel)
  using indRelRTPO.trans
  unfolding trans-def
  by blast
ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ preorder Rel
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
  unfolding preorder-on-def
  by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ preorder Rel
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
from this obtain Rel where B1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and B2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
and B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel and B4: preorder Rel
and B5: weak-reduction-bisimulation Rel (STCal Source Target)
  by blast
from B2 B4 have B6: refl TRel
  unfolding preorder-on-def refl-on-def
  by blast
from B2 B4 have B7: trans TRel
  unfolding trans-def preorder-on-def
  by blast
hence B8: TRel^+ = TRel
  using trancl-id[of TRel]
  by simp
with B6 have B9: TRel^+ = TRel
  using refl-trancl[of TRel]
  unfolding refl-on-def
  by blast
with B3 have ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^+
  by simp
moreover from B2 B8 have ∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel
and ∨ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel^+
  by auto
ultimately have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ T1 T2. (T1, T2) ∈ TRel → (TargetTerm T1, TargetTerm T2) ∈ Rel)
  ∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel → (T1, T2) ∈ TRel^+)
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel^+)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
  using B1 B5
  by blast
hence operational-corresponding (TRel^+)
  ∧ weak-reduction-bisimulation (TRel^+) Target
proof
(lemma qed auto)
proof
(weak-reduction-bisimulation)
show
(assumes assume TRel fixes ultimately show operational-corresponding TRel
moreover from B6 B7 have preorder TRel unfolding preorder-on-def
ultimately show operational-corresponding TRel \land preorder TRel
\land weak-reduction-bisimulation TRel Target
by blast
qed

lemma (in encoding) OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation:
fixes TRel :: ('proc T \times 'proc T) set
assumes eqT: equivalence TRel
shows (operational-corresponding TRel \land weak-reduction-bisimulation TRel Target) \iff
weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
proof (rule iffI, erule conjE)
assume oc: operational-corresponding TRel and bisimT: weak-reduction-bisimulation TRel Target
show weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
proof auto
fix P Q P'
assume P ~[]T TRel> Q and P \longrightarrow (STCal Source Target)* P'
thus \exists Q'. Q \longrightarrow (STCal Source Target)* Q' \land P' ~[]T TRel> Q'
proof (induct arbitrary: P')
case (encR S)
assume SourceTerm S \longrightarrow (STCal Source Target)* P'
from this obtain S' where A1: S \longrightarrow Source* S' and A2: S' \in S P'
  by (auto simp add: STCal-steps(1))
from A1 oc obtain T where A3: [S] \longrightarrow Target* T and A4: ([S'], T) \in T Rel
  by blast
from A3 have TargetTerm ([S]) \longrightarrow (STCal Source Target)* (TargetTerm T)
  by (simp add: STCal-steps(2))
moreover have P' ~[]T TRel> TargetTerm T
proof
  from A2 have P' ~[]T TRel> TargetTerm ([S'])
    by (simp add: indRelTEQ.encR)
  moreover from A4 have TargetTerm ([S']) ~[]T TRel> TargetTerm T
    by (rule indRelTEQ.target)
  ultimately show P' ~[]T TRel> TargetTerm T
    by (rule indRelTEQ.trans)
qed
ultimately show \exists Q'. TargetTerm ([S]) \longrightarrow (STCal Source Target)* Q' \land P' ~[]T TRel> Q'
  by blast
next
case (encL S)
assume TargetTerm ([S]) \longrightarrow (STCal Source Target)* P'
from this obtain T where B1: [S] \longrightarrow Target* T and B2: T \in T P'
  by (auto simp add: STCal-steps(2))
from B1 oc obtain S' where B3: S \longrightarrow Source* S' and B4: ([S'], T) \in T Rel
  by blast
from B3 have SourceTerm S \longrightarrow (STCal Source Target)* (SourceTerm S')
  by (simp add: STCal-steps(1))
moreover have P' ~[]T TRel> SourceTerm S'
proof
  from B4 eqT have (T, [S']) \in T Rel
    unfolding equiv-def sym-def
    by blast
  with B2 have P' ~[]T TRel> TargetTerm ([S'])
    by (simp add: indRelTEQ.target)
  moreover have TargetTerm ([S']) ~[]T TRel> SourceTerm S'
    by blast
next
  case (trans P Q R)
  assume P \rightarrow \STCal \SourceTerm \TargetTerm \* P'
  \land P', P \rightarrow \STCal \SourceTerm \TargetTerm \* P'
  \rightarrow \exists Q'. Q \rightarrow \STCal \SourceTerm \TargetTerm \* Q' \land P' \sim \[ \] T < TRel > Q'
  from this obtain Q' where D1: Q \rightarrow \STCal \SourceTerm \TargetTerm \* Q' \land D2: P' \sim \[ \] T < TRel > Q'
  by blast
  assume \bigwedge Q'. Q \rightarrow \STCal \SourceTerm \TargetTerm \* Q'
  \rightarrow \exists R'. R \rightarrow \STCal \SourceTerm \TargetTerm \* R' \land Q' \sim \[ \] T < TRel > R'
  with D1 obtain R' where D3: R \rightarrow \STCal \SourceTerm \TargetTerm \* R' \land D4: Q' \sim \[ \] T < TRel > R'
  by blast
  from D2 D4 have P' \sim \[ \] T < TRel > R'
  by (rule indRelTEQ.trans)
  with D3 show \exists R'. R \rightarrow \STCal \SourceTerm \TargetTerm \* R' \land P' \sim \[ \] T < TRel > R'
  by blast
qed
next
fix P Q Q'
assume P \sim \[ \] T < TRel > Q and Q \rightarrow \STCal \SourceTerm \TargetTerm \* Q'
thus \exists P'. P \rightarrow \STCal \SourceTerm \TargetTerm \* P' \land P' \sim \[ \] T < TRel > Q'
proof (induct arbitrary: Q')
  case (encR S)
  assume TargetTerm \ ([S]) \rightarrow \STCal \SourceTerm \TargetTerm \* Q'
  from this obtain T where E1: [S] \rightarrow \TargetTerm \* T and E2: T \in T Q'
  by (auto simp add: STCal-steps(2))
  from E1 oc obtain S' where E3: S \rightarrow \SourceTerm \* S' \land E4: ([S'], T) \in TRel
  by blast
  from E3 have SourceTerm S \rightarrow \STCal \SourceTerm \TargetTerm \* (SourceTerm S')
  by (simp add: STCal-steps(1))
  moreover have SourceTerm S' \sim \[ \] T < TRel > Q'
  proof
    have SourceTerm S' \sim \[ \] T < TRel > TargetTerm \ ([S'])
    by (rule indRelTEQ.encR)
    moreover from E2 E4 have TargetTerm \ ([S']) \sim \[ \] T < TRel > Q'
    by (simp add: indRelTEQ.target)
    ultimately show SourceTerm S' \sim \[ \] T < TRel > Q'
    by (rule indRelTEQ.trans)
  qed
  ultimately show \exists P'. SourceTerm S \rightarrow \STCal \SourceTerm \TargetTerm \* P' \land P' \sim \[ \] T < TRel > Q'
  by blast
next
case (encL S)
assume SourceTerm S \mapsto (STCal Source Target) \ast Q'
from this obtain S' where F1: S \mapsto Source \ast S' and F2: S' \in S Q'
  by (auto simp add: STCal-steps(1))
from F1 oc obtain T where F3: [S] \mapsto Target \ast T and F4: ([S'], T) \in TRel
  by blast
from F3 have TargetTerm ([S]) \mapsto (STCal Source Target) \ast (TargetTerm T)
  by (simp add: STCal-steps(2))
moreover have TargetTerm T \sim [] T \langle TRel \rangle Q'
  proof
    from F4 eqT have (T, [S']) \in TRel
      unfolding equiv-def sym-def
      by blast
    hence TargetTerm T \sim [] T \langle TRel \rangle TargetTerm ([S'])
      by (rule indRelTEQ.target)
    moreover from F2 have TargetTerm ([S']) \sim [] T \langle TRel \rangle Q'
      by (simp add: indRelTEQ.encL)
    ultimately show TargetTerm T \sim [] T \langle TRel \rangle Q'
      by (rule indRelTEQ.trans)
    qed
  ultimately show \exists P'. TargetTerm ([S']) \mapsto (STCal Source Target) \ast P' \land P' \sim [] T \langle TRel \rangle Q'
    by blast
next
case (target T1 T2)
assume TargetTerm T2 \mapsto (STCal Source Target) \ast Q'
from this obtain T2' where G1: T2 \mapsto Target \ast T2' and G2: T2' \in T Q'
  by (auto simp add: STCal-steps(2))
assume (T1', T2') \in TRel
with G1 bisimT obtain T1' where G3: T1 \mapsto Target \ast T1' and G4: (T1', T2') \in TRel
  by blast
from G3 have TargetTerm T1 \mapsto (STCal Source Target) \ast (TargetTerm T1')
  by (simp add: STCal-steps(2))
moreover from G2 G4 have TargetTerm T1' \sim [] T \langle TRel \rangle Q'
  by (simp add: indRelTEQ.target)
ultimately show \exists P'. TargetTerm T1 \mapsto (STCal Source Target) \ast P' \land P' \sim [] T \langle TRel \rangle Q'
  by blast
next
case (trans P Q R R')
assume R \mapsto (STCal Source Target) \ast R'
  and \[ R', R \mapsto (STCal Source Target) \ast R' \]
  \implies \exists Q'. Q \mapsto (STCal Source Target) \ast Q' \land Q' \sim [] T \langle TRel \rangle R'
from this obtain Q' where H1: Q \mapsto (STCal Source Target) \ast Q' and H2: Q' \sim [] T \langle TRel \rangle R'
  by blast
assume \[ Q', Q \mapsto (STCal Source Target) \ast Q' \]
  \implies \exists P'. P \mapsto (STCal Source Target) \ast P' \land P' \sim [] T \langle TRel \rangle Q'
with H1 obtain P' where H3: P \mapsto (STCal Source Target) \ast P' and H4: P' \sim [] T \langle TRel \rangle Q'
  by blast
from H4 H2 have P' \sim [] T \langle TRel \rangle R'
  by (rule indRelTEQ.trans)
with H3 show \exists P'. P \mapsto (STCal Source Target) \ast P' \land P' \sim [] T \langle TRel \rangle R'
  by blast
qed
qed
next
assume bisim: weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
have operational-corresponding TRel
proof auto
fix S S'
have SourceTerm S \sim [] T \langle TRel \rangle TargetTerm ([S])
  by (rule indRelTEQ.encR)
moreover assume S \mapsto Source \ast S'

187
hence SourceTerm S \mapsto (STCal Source Target)* (SourceTerm S')
by (simp add: STCal-steps(1))
ultimately obtain Q' where I1: TargetTerm ([S]) \mapsto (STCal Source Target)* Q'
and I2: SourceTerm S \sim\!
\begin{align*}
\end{align*}
next
proof
weak-reduction-bisimulation TRel Target
moreover have \begin{align*}
\end{align*}
ultimately show \begin{align*}
\end{align*}
qed
moreover have weak-reduction-bisimulation TRel Target
proof -
from eqT have TRel" = TRel
using reflcl-trancl[of TRel] trancl-id[of TRel]
unfolding equiv-def refl-on-def
by auto
with J2 J4 have ([S'], T) \in TRel
using indRelTEQ-to-TRel(2)[where TRel=TRel]
trans-closure-of-TRel-refl-cond[where TRel=TRel]
by simp
with J3 show \exists T. ([S] \implies Target* T \land ([S'], T) \in TRel
by blast
qed

lemma (in encoding) OC-wrt-equivalence-iff-weak-reduction-bisimulation:
fixes TRel :: (‘procT x ‘procT) set
assumes eqT: equivalence TRel
shows (operational-corresponding TRel \land weak-reduction-bisimulation TRel Target) \iff \exists Rel.
(\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \land (TargetTerm ([S]), SourceTerm S) \in Rel)
\( \land \) \( T \in \{(T_1, T_2), (\text{TargetTerm} \ T_1, \text{TargetTerm} \ T_2) \in \text{Rel}\} \)
\( \land \) \( \text{trans} \ \text{Rel} \land \text{weak-reduction-bisimulation} \ \text{Rel} \ (\text{STCal Source Target}) \)

\textbf{proof (rule iff1, erule conjE)}

\textbf{assume \( \text{oc} \): operational-corresponding \( T \in \text{Rel} \) \text{and} \( \text{bisimT}: \text{weak-reduction-bisimulation} \ T \in \text{Rel} \) } \text{Target}

\textbf{from eqT have \( rt: T \in \text{Rel}^* = T \in \text{Rel} \) }
\begin{itemize}
  \item \text{using refl-cl-trancl[of TRel] trancl-id[of TRel]}
\end{itemize}

\textbf{unfolding equiv-def refl-on-def}
\textbf{by auto}

\textbf{have \( \exists S. \text{SourceTerm} \ S \sim -<T T \in \text{Rel}> \text{TargetTerm} \ ([S]) \land \text{TargetTerm} \ ([S]) \sim -<T T \in \text{Rel}> \text{SourceTerm} \ S \) }
\textbf{by (simp add: indRelTEQ.encR encR indRelTEQ.encL)}

\textbf{moreover from \( rt \) have \( T \in \text{Rel} = \{(T_1, T_2), \text{TargetTerm} T_1 \sim -<T T \in \text{Rel}> \text{TargetTerm} T_2 \) }
\begin{itemize}
  \item \text{using indRelTEQ-to-TRel(4)[where TRel=TRel]}
  \item \text{trans-closure-of-TRel-refl-cond[where TRel=TRel]}
\end{itemize}
\textbf{by (auto simp add: indRelTEQ.target)}

\textbf{moreover have \( \text{trans (indRelTEQ TRel)} \) }
\textbf{using indRelTEQ,trans[where TRel=TRel]}
\textbf{unfolding trans-def}
\textbf{by blast}

\textbf{moreover from eqT \( \text{oc bisimT} \) }

\textbf{have \( \text{weak-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)} \) }
\textbf{using OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation[where TRel=TRel]}
\textbf{by blast}

\textbf{ultimately show \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\text{TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel}) \land \text{trans Rel} \land \text{weak-reduction-bisimulation} \text{Rel} \ (\text{STCal Source Target}) \) }
\textbf{by blast}

\textbf{next}

\textbf{assume \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\text{TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel}) \land \text{trans Rel} \land \text{weak-reduction-bisimulation} \text{Rel} \ (\text{STCal Source Target}) \) }

\textbf{from this obtain \( \text{Rel where A1: } \forall S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\text{TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel} \land \text{trans Rel} \land \text{weak-reduction-bisimulation} \text{Rel} \ (\text{STCal Source Target}) \) }
\textbf{and \( A2: T \in \text{Rel} = \{(T_1, T_2), \text{TargetTerm} T_1, \text{TargetTerm} T_2 \in \text{Rel}\} \) and \( A3: \text{trans Rel} \) }
\textbf{and \( A4: \text{weak-reduction-bisimulation} \text{Rel} \ (\text{STCal Source Target}) \) }
\textbf{by blast}

\textbf{have operational-corresponding \( T \in \text{Rel} \) }
\textbf{proof auto}

\textbf{fix \( S S' \) }
\textbf{from A1 have \( (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \) }
\textbf{by simp}

\textbf{moreover assume \( S \mapsto \text{Source} \ S' \) }

\textbf{hence \( \text{SourceTerm} \mapsto (\text{STCal Source Target})* (\text{SourceTerm} \ S') \) }
\textbf{by (simp add: STCal-steps(1))}

\textbf{ultimately obtain \( Q \) where B1: \( \text{TargetTerm} \ ([S]) \mapsto (\text{STCal Source Target})* Q' \) }
\textbf{and B2: \( \text{SourceTerm} \ S', Q' \in \text{Rel} \) }
\textbf{using A4}
\textbf{by blast}

\textbf{from B1 obtain \( T \) where B3: \( [S'] \mapsto \text{Target} \* T \) and B4: \( T \in T \in Q' \) }
\textbf{by (auto simp add: STCal-steps(2))}

\textbf{from A1 have \( (\text{TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel} \) }
\textbf{by simp}

\textbf{with B2 A3 have \( (\text{TargetTerm} \ ([S]), Q') \in \text{Rel} \) }
\textbf{unfolding trans-def}
\textbf{by blast}

\textbf{with B4 A2 have \( ([S'], T) \in \text{TRel} \) }
\textbf{by simp}

\textbf{with B3 show \( \exists T. [S] \mapsto \text{Target} \* T \land ([S'], T) \in \text{TRel} \) }
\textbf{by blast}
next
fix $S$ $T$
from $A1$ have $(\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}$
  by simp
moreover assume $[S] \mapsto \text{Target} \ast T$
hence $\text{TargetTerm } ([S]) \mapsto (\text{STCal Source Target}) \ast (\text{TargetTerm } T)$
  by (simp add: STCal-steps(2))
ultimately obtain $P'$ where $C1$: $\text{SourceTerm } S \mapsto (\text{STCal Source Target}) \ast P'$
  and $C2$: $(P', \text{TargetTerm } T) \in \text{Rel}$
  using $A4$
  by blast
from $C1$ obtain $S'$ where $C3$: $S \mapsto \text{Source} \ast S'$ and $C4$: $S' \in S P'$
  by (auto simp add: STCal-steps(1))
from $A1$ $C4$ have $\text{TargetTerm } ([S']), P') \in \text{Rel}$
  by simp
from $A3$ this $C2$ have $(\text{TargetTerm } ([S']), \text{TargetTerm } T) \in \text{Rel}$
  unfolding trans-def
  by blast
with $A2$ have $([S'], T) \in T\text{Rel}$
  by simp
with $C3$ show $\exists S', S \mapsto \text{Source} \ast S' \land ([S'], T) \in T\text{Rel}$
  by blast
qed
moreover have weak-reduction-bisimulation $T\text{Rel} \text{Target}$
proof
  auto
fix $TP$ $TQ$ $TP'$
assumption $(TP, TQ) \in T\text{Rel}$
with $A2$ have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$
  by simp
moreover assume $TP \mapsto \text{Target} \ast TP'$
hence $\text{TargetTerm } TP \mapsto (\text{STCal Source Target}) \ast (\text{TargetTerm } TP')$
  by (simp add: STCal-steps(2))
ultimately obtain $Q'$ where $D1$: $\text{TargetTerm } TQ \mapsto (\text{STCal Source Target}) \ast Q'$
  and $D2$: $(\text{TargetTerm } TP', Q') \in \text{Rel}$
  using $A4$
  by blast
from $D1$ obtain $TQ'$ where $D3$: $TQ \mapsto \text{Target} \ast TQ'$ and $D4$: $TQ' \in T Q'$
  by (auto simp add: STCal-steps(2))
from $A2$ $D2$ $D4$ have $(TP', TQ') \in T\text{Rel}$
  by simp
with $D3$ show $\exists TQ'. TQ \mapsto \text{Target} \ast TQ' \land (TP', TQ') \in T\text{Rel}$
  by blast
next
fix $TP$ $TQ$ $TP'$
assumption $(TP, TQ) \in T\text{Rel}$
with $A2$ have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$
  by simp
moreover assume $TP \mapsto \text{Target} \ast TP'$
hence $\text{TargetTerm } TP \mapsto (\text{STCal Source Target}) \ast (\text{TargetTerm } TP')$
  by (simp add: STCal-steps(2))
ultimately obtain $P'$ where $E1$: $\text{TargetTerm } TP \mapsto (\text{STCal Source Target}) \ast P'$
  and $E2$: $(P', \text{TargetTerm } TQ') \in \text{Rel}$
  using $A4$
  by blast
from $E1$ obtain $TP'$ where $E3$: $TP \mapsto \text{Target} \ast TP'$ and $E4$: $TP' \in T P'$
  by (auto simp add: STCal-steps(2))
from $A2$ $E2$ $E4$ have $(TP', TQ') \in T\text{Rel}$
  by simp
with $E3$ show $\exists TP'. TP \mapsto \text{Target} \ast TP' \land (TP', TQ') \in T\text{Rel}$
  by blast
qed
ultimately show operational-corresponding $\text{TRel} \land \text{weak-reduction-bisimulation} \ \text{TRel} \ \text{Target}$

by simp

qed

An encoding is strong operational corresponding w.r.t a strong bisimulation on target terms $\text{TRel}$ iff there exists a relation, like $\text{indRelRTPO}$, that relates at least all source terms and their literal translations, includes $\text{TRel}$, 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** $\text{TRel} :: (\text{\texttt{\textnormal{procT} \times procT}}) \ \text{set}$
- **shows** $(\text{strongly-operational-corresponding} \ (\text{TRel}^*) \ \land \ \text{strong-reduction-bisimulation} \ (\text{TRel}^*) \ \text{Target})$
  
  $\Rightarrow$ $\text{strong-reduction-bisimulation} \ (\text{indRelRTPO} \ \text{TRel}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

**proof** (rule iffI, erule conjE)

- **assume** $\text{ocurr: strongly-operational-corresponding} \ (\text{TRel}^*)$
  
  **and** $\text{bisim: strong-reduction-bisimulation} \ (\text{TRel}^*) \ \text{Target}$

- **hence** $\text{strong-reduction-simulation} \ (\text{indRelRTPO} \ \text{TRel}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

- **using** SOCom-iff-indRelRTPO-is-strong-reduction-simulation[where $\text{TRel= TRel}$]

  by simp

- **moreover from** $\text{bisim have} \ \text{strong-reduction-simulation} \ ((\text{TRel}^*)^{-1}) \ \text{Target}$

  **using** strong-reduction-bisimulations-impl-inverse-is-simulation[where $\text{Rel}= \text{TRel}^*$]

  by simp

- **with** $\text{ocurr}$

  **have** $\text{strong-reduction-simulation} \ ((\text{indRelRTPO} \ \text{TRel})^{-1}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

  **using** SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation[where $\text{TRel}= \text{TRel}$]

  by simp

- **ultimately show** $\text{strong-reduction-bisimulation} \ (\text{indRelRTPO} \ \text{TRel}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

  **using** strong-reduction-simulations-impl-bisimulation[where $\text{Rel}= \text{indRelRTPO} \ \text{TRel}$]

  by simp

- **next**

- **assume** $\text{bisim: strong-reduction-bisimulation} \ (\text{indRelRTPO} \ \text{TRel}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

- **hence** $\text{strongly-operational-complete} \ (\text{TRel}^*) \ \land \ \text{strong-reduction-simulation} \ (\text{TRel}^*) \ \text{Target}$

  **using** SOCom-iff-indRelRTPO-is-strong-reduction-simulation[where $\text{TRel= TRel}$]

  by simp

- **moreover from** $\text{bisim}$

  **have** $\text{strong-reduction-simulation} \ ((\text{indRelRTPO} \ \text{TRel})^{-1}) \ (\text{STCal} \ \text{Source} \ \text{Target})$

  **using** strong-reduction-bisimulations-impl-inverse-is-simulation[where $\text{Rel}= \text{indRelRTPO} \ \text{TRel}$]

  by simp

- **hence** $\text{strongly-operational-sound} \ (\text{TRel}^*) \ \land \ \text{strong-reduction-simulation} \ ((\text{TRel}^*)^{-1}) \ \text{Target}$

  **using** SOSou-iff-inverse-of-indRelRTPO-is-strong-reduction-simulation[where $\text{TRel}= \text{TRel}$]

  by simp

- **ultimately show** $\text{strongly-operational-corresponding} \ (\text{TRel}^*)$

  **and** $\text{strong-reduction-bisimulation} \ (\text{TRel}^*) \ \text{Target}$

  **using** strong-reduction-simulations-impl-bisimulation[where $\text{Rel}= \text{TRel}^*$]

  by simp

qd

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

- **fixes** $\text{TRel} :: (\text{\texttt{\textnormal{procT} \times procT}}) \ \text{set}$

- **shows** $(\text{strongly-operational-corresponding} \ (\text{TRel}^*) \ \land \ \text{strong-reduction-bisimulation} \ (\text{TRel}^*) \ \text{Target})$

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

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

  $\land \ (\forall \ \text{T1} \ \text{T2}. \ (\text{TargetTerm} \ \text{T1}, \ \text{TargetTerm} \ \text{T2}) \in \text{Rel} \ \rightarrow \ (\text{T1}, \ \text{T2}) \in \text{TRel}^*)$

  $\land \ (\forall \ \text{S} \ . \ (\text{SourceTerm} \ \text{S}, \ \text{TargetTerm} \ \text{T}) \in \text{Rel} \ \rightarrow \ (\text{[S]}, \ \text{T}) \in \text{TRel}^*)$

  $\land \ \text{strong-reduction-bisimulation} \ \text{Rel} \ (\text{STCal} \ \text{Source} \ \text{Target})$)

**proof** (rule iffI, erule conjE)

- **have** $\forall \ \text{S}. \ (\text{SourceTerm} \ \text{S}, \ \text{TargetTerm} \ ((\text{S}))) \in \text{indRelRTPO} \ \text{TRel}$

  **by** simp add: indRelRTPO.encR

- **moreover have** $\forall \ \text{T1} \ \text{T2}. \ (\text{T1}, \ \text{T2}) \in \text{TRel} \ \rightarrow \ \text{TargetTerm} \ \text{T1} \ \less \ 1 ] \ \text{RT} < \text{TRel} > \ \text{TargetTerm} \ \text{T2}$
by (simp add: indRelRTPO.target)
moreover have \(\forall T_1 \ T_2. \ \text{TargetTerm} \ T_1 \triangleq \llbracket \text{TTerm} < \text{TRel} > \rrbracket \ \text{TargetTerm} \ T_2 \longrightarrow (T_1, T_2) \in \text{TRel}^+\)
using indRelRTPO-to-TRel(4)[where \(\text{TRel} = \text{TRel}\)]
by simp
moreover have \(\forall S \ T. \ \text{SourceTerm} \ S \triangleq \llbracket \text{TTerm} < \text{TRel} > \rrbracket \ \text{TargetTerm} \ T \longrightarrow ([S], T) \in \text{TRel}^\ast\)
using indRelRTPO-to-TRel(2)[where \(\text{TRel} = \text{TRel}\)]
trans-closure-of-TRel-refl-cond
by simp
moreover assume strongly-operational-corresponding \((\text{TRel}^\ast)\)
and strongly-reduction-bisimulation \((\text{TRel}^+\ast)\) \text{Target}

hence strongly-reduction-bisimulation \((\text{indRelRTPO} \ \text{TRel})\) \((\text{STCal Source Target})\)
using \(\text{SOC-iff-indRelRTPO}\)-is-strong-reduction-bisimulation\[where \(\text{TRel} = \text{TRel}\)]
by simp
ultimately show \(\exists \text{Rel}. \ (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel})\)
\(\land \ (\forall T_1 \ T_2. \ (T_1, T_2) \in \text{TRel} \longrightarrow (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel})\)
\(\land \ (\forall T_1 \ T_2. \ (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel} \longrightarrow (T_1, T_2) \in \text{TRel}^+\)
\(\land \ (\forall S \ T. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
\(\land \ \text{strong-reduction-bisimulation Rel} \ (\text{STCal Source Target})\)

by blast

next
assume \(\exists \text{Rel}. \ (\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel})\)
\(\land \ (\forall T_1 \ T_2. \ (T_1, T_2) \in \text{TRel} \longrightarrow (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel})\)
\(\land \ (\forall T_1 \ T_2. \ (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel} \longrightarrow (T_1, T_2) \in \text{TRel}^+\)
\(\land \ (\forall S \ T. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
\(\land \ \text{strong-reduction-bisimulation Rel} \ (\text{STCal Source Target})\)

from this obtain Rel where A1: \(\forall S. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} \ ([S])) \in \text{Rel}\)
and A2: \(\forall T_1 \ T_2. \ (T_1, T_2) \in \text{TRel} \longrightarrow (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel}\)
and A3: \(\forall T_1 \ T_2. \ (\text{TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel} \longrightarrow (T_1, T_2) \in \text{TRel}^+\)
and A4: \(\forall S \ T. \ (\text{SourceTerm} \ S, \ \text{TargetTerm} T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
and A5: \(\text{strong-reduction-bisimulation Rel} \ (\text{STCal Source Target})\)

by blast

hence strongly-operational-complete \((\text{TRel}^\ast)\)
\(\land \ \text{strong-reduction-simulation} \ (\text{TRel}^+\ast) \ \text{Target}\)
using \(\text{SOCCom-iff-strong-reduction-simulation}\)[where \(\text{TRel} = \text{TRel}\)]
by blast

moreover from A5 have strong-reduction-simulation \((\text{Rel}^{-1})\) \((\text{STCal Source Target})\)
using \(\text{strong-reduction-bisimulations-impl-inverse-is-simulation}\)[where \(\text{Rel} = \text{Rel}\)]
by simp

with A1 A2 A3 A4 have strongly-operational-sound \((\text{TRel}^\ast)\)
\(\land \ \text{strong-reduction-simulation} \ ((\text{TRel}^+\ast)^{-1}) \ \text{Target}\)
using \(\text{SOSow-iff-strong-reduction-simulation}\)[where \(\text{TRel} = \text{TRel}\)]
by simp

ultimately show strongly-operational-corresponding \((\text{TRel}^\ast)\)
\(\land \ \text{strong-reduction-bisimulation} \ (\text{TRel}^+\ast) \ \text{Target}\)
using \(\text{strong-reduction-simulations-impl-bisimulation}\)[where \(\text{Rel} = \text{TRel}^\ast\)]
by simp

qed

lemma \((\text{in encoding})\) \(\text{SOC-wrt-preorder-iff-strong-reduction-bisimulation}:\)
fixes TRel :: \(\langle \text{proc} T \times \text{proc} T \rangle\) set
shows \((\text{strongly-operational-corresponding} \ \text{TRel} \ \land \ \text{preorder} \ \text{TRel})\)
\(\land \ \text{strong-reduction-bisimulation} \ \text{TRel} \ \text{Target}\)
= \((\exists \text{Rel}. \ (\forall S. \ (\text{SourceTerm} S, \ \text{TargetTerm} \ ([S])) \in \text{Rel})\)
\(\land \ \text{TRel} = \{(T_1, T_2. \ \text{(TargetTerm} T_1, \ \text{TargetTerm} T_2) \in \text{Rel}\}\)
\(\land \ (\forall S \ T. \ (\text{SourceTerm} S, \ \text{TargetTerm} T) \in \text{Rel} \longrightarrow ([S], T) \in \text{TRel}^\ast\)
\(\land \ \text{preorder Rel}\)
\(\land \ \text{strong-reduction-bisimulation Rel} \ (\text{STCal Source Target})\))

proof \(\text{rule iffI, erule conjE, erule conjE, erule conjE}\)

assume A1: strongly-operational-complete \text{TRel} \ \text{and A2: strongly-operational-sound} \text{TRel}
and A3: preorder \text{TRel} \ \text{and A4: strongly-reduction-bisimulation} \text{TRel} \ \text{Target}

from A3 have A5: \(\text{TRel}^+ = \text{TRel}\)
using \(\text{trancl-id}[\text{of TRel}]\)

192
unfolding preorder-on-def
by blast
with A3 have TRel* = TRel
  unfolding preorder-on-def refl-on-def
by blast
with A1 A2 have strongly-operational-corresponding (TRel*)
by simp
moreover from A4 A5 have strong-reduction-bisimulation (TRel+)
proof auto
fix TP TQ
assume (TP, TQ) ∈ TRel
thus TargetTerm TP ≲ TRel TargetTerm TQ
by (rule indRelRTPO.target)
next
fix TP TQ
assume TargetTerm TP ≲ [\_\_] RT<TRel> TargetTerm TQ
with A3 show (TP, TQ) ∈ TRel
  unfolding preorder-on-def refl-on-def
by blast
with A3 have ∀ S T. (SourceTerm S, TargetTerm ([S])) ∈ indRelRTPO TRel
  unfolding preorder-on-def
by blast
moreover from A3
have ∀ S T. (SourceTerm S, TargetTerm ([S])) ∈ indRelRTPO TRel → ([S], T) ∈ TRel+
  unfolding preorder-on-def refl-on-def
by blast
with A3 have ∀ S T. (SourceTerm S, TargetTerm ([S])) ∈ indRelRTPO TRel
  unfolding preorder-on-def
by blast
moreover from A3 have refl (indRelRTPO TRel)
  unfolding preorder-on-def
by (simp add: indRelRTPO-refl)
moreover have trans (indRelRTPO TRel)
  unfolding preorder-on-def
by blast
ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel)
  ∧ TRel = { (T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel }
  ∧ (∀ S T. (SourceTerm S, TargetTerm ([S])) ∈ Rel → ([S], T) ∈ TRel)
  ∧ preorder Rel
  ∧ strong-reduction-bisimulation Rel (STCal Source Target)
unfolding preorder-on-def
by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S]))) ∈ Rel)
  ∧ TRel = { (T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel }
  ∧ (∀ S T. (SourceTerm S, TargetTerm ([S])) ∈ Rel → ([S], T) ∈ TRel)
  ∧ preorder Rel
  ∧ strong-reduction-bisimulation Rel (STCal Source Target)
from this obtain \( Rel \) where \( B1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel \)
and \( B2: \text{TRel} = \{ (T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \} \)
and \( B3: \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \rightarrow ([S], T) \in TRel \) and \( B4: \text{preorder } Rel \)
and \( B5: \text{strong-reduction-bisimulation } Rel \) (STCal Source Target)

by blast
from \( B2 B4 \) have \( B6: \text{refl } TRel \)
unfolding preorder-on-def refl-on-def
by blast
from \( B2 B4 \) have \( B7: \text{trans } TRel \)
unfolding trans-def preorder-on-def
by blast
hence \( B8: \text{TRel}^+ \equiv TRel \)
by (rule trancl-id)
with \( B6 \) have \( B9: \text{TRel}^* \equiv TRel \)
unfolding refl-on-def
by blast
with \( B3 \) have \( \forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \rightarrow ([S], T) \in TRel^* \)
by simp
moreover from \( B2 B8 \) have \( \forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \)
and \( \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \rightarrow (T1, T2) \in TRel^+ \)
by auto
ultimately have \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \)
\wedge (\forall T1 T2. (T1, T2) \in TRel \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel)
\wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel \rightarrow (T1, T2) \in TRel^+)
\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel \rightarrow ([S], T) \in TRel^*)
\wedge \text{strong-reduction-bisimulation } Rel \) (STCal Source Target)
using \( B1 B5 \)
by blast
hence \( \text{strongly-operational-corresponding } (TRel^*) \wedge \text{strong-reduction-bisimulation } (TRel^+) \) Target
using SOC-iff-strong-reduction-bisimulation[where TRel=TRel]
by simp
with \( B8 B9 \) have \( \text{strongly-operational-corresponding } TRel \wedge \text{strong-reduction-bisimulation } TRel \) Target
by simp
moreover from \( B6 B7 \) have \( \text{preorder } TRel \)
unfolding preorder-on-def
by blast
ultimately show \( \text{strongly-operational-corresponding } TRel \wedge \text{preorder } TRel \)
\wedge \text{strong-reduction-bisimulation } TRel \) Target
by blast
qed

lemma (in encoding) SOC-wrt-TRel-iff-strong-reduction-bisimulation:
shows \( (\exists \text{TRel. strongly-operational-corresponding } (TRel^*) \)
\wedge \text{strong-reduction-bisimulation } (TRel^+) \) Target
\( = (\exists \text{Rel. } \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel) \)
\wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in Rel
\rightarrow (\text{TargetTerm } ([S]), \text{TargetTerm } T) \in Rel^*)
\wedge \text{strong-reduction-bisimulation } Rel \) (STCal Source Target)

proof (rule iffI)
assume \( \exists \text{TRel. strongly-operational-corresponding } (TRel^*) \)
\wedge \text{strong-reduction-bisimulation } (TRel^+) \) Target
from this obtain \( TRel \) where \( \text{strongly-operational-corresponding } (TRel^*) \)
and \( \text{strong-reduction-bisimulation } (TRel^+) \) Target
by blast
hence \( \text{strong-reduction-bisimulation } (\text{indRelRTPO } TRel) \) (STCal Source Target)
using SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]
by simp
moreover have \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{indRelRTPO } TRel \)
by (simp add: indRelRTPO.encR)
moreover have \( \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in (\text{indRelRTPO} \ TRel) \nRightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in (\text{indRelRTPO} \ TRel) \) 

using \( \text{indRelRTPO-relates-source-target}[\text{where} \ TRel=\text{TRel}] \)

by simp

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

\( \wedge (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \nRightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}^+) \) 

\( \wedge \text{strong-reduction-bisimulation} \) 

\( \text{Rel} \) 

(\text{STCal Source Target}) 

by \( \text{blast} \)

next

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

\( \wedge (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \nRightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}^+) \) 

\( \wedge \text{strong-reduction-bisimulation} \) 

\( \text{Rel} \) 

(\text{STCal Source Target}) 

from this obtain \( \text{Rel} \) where \( A1: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \)

and \( A2: \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \nRightarrow (\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}^+ \)

and \( A3: \text{strong-reduction-bisimulation} \) 

\( \text{Rel} \) 

(\text{STCal Source Target}) 

by \( \text{blast} \)

from \( A2 \) obtain \( \text{TRel} \) where \( \forall T1 T2. (T1, T2) \in \text{TRel} \nRightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \)

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

and \( \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \nRightarrow ([S], T) \in \text{TRel}^+ \)

using \( \text{target-relation-from-source-target-relation}[\text{where} \ Rel=\text{Rel}] \)

by \( \text{simp} \)

with \( A1 A3 \) have \( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \)

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

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

\( \wedge (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \nRightarrow ([S], T) \in \text{TRel}^+) \)

\( \wedge \text{strong-reduction-bisimulation} \) 

\( \text{Rel} \) 

(\text{STCal Source Target}) 

by \( \text{blast} \)

hence \( \text{strongly-operational-corresponding} \) 

(\text{TRel}^+) 

\( \wedge \text{strong-reduction-bisimulation} \) 

(\text{TRel}^+) 

Target 

using \( \text{SOC-iff-strong-reduction-bisimulation}[\text{where} \ TRel=\text{TRel}] \)

by simp

thus \( \exists \text{TRel} \) 

\( \text{strongly-operational-corresponding} \) 

(\text{TRel}^+) 

\( \wedge \text{strong-reduction-bisimulation} \) 

(\text{TRel}^+) 

Target 

by \( \text{blast} \)

qed

lemma (in encoding) \( \text{SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation}: \)

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

\( \text{assumes} \ eqT; \text{equivalence} \ TRel \)

\( \text{shows} \ (\text{strongly-operational-corresponding} \ TRel \wedge \text{strong-reduction-bisimulation} \ TRel \) 

Target 

\( \text{\textit{\lhd} \text{strong-reduction-bisimulation} \ (indRelTEQ} \) 

(\text{TRel}) \) 

(\text{STCal Source Target}) 

\( \text{proof} \) (rule \( \text{iffI, erule conjE} \))

\( \text{assume oc; \text{strongly-operational-corresponding} TRel} \)

\( \text{and bisimT; \text{strong-reduction-bisimulation} TRel Target} \)

\( \text{show} \ \text{strong-reduction-bisimulation} \) 

(\text{indRelTEQ} \) 

(\text{TRel}) 

(\text{STCal Source Target}) 

\( \text{proof} \) auto

fix \( P \) \( Q \) \( P' \)

assume \( P \sim^T [T<\text{TRel}>] Q \) and \( P \nRightarrow (\text{STCal Source Target}) P' \)

thus \( \exists Q'. Q \nRightarrow (\text{STCal Source Target}) Q' \wedge P' \sim^T [T<\text{TRel}>] Q' \)

\( \text{proof} \) (induct arbitrary: \( P' \))

\( \text{case} (\text{enc1 S}) \)

\( \text{assume} \ SourceTerm S \nRightarrow (\text{STCal Source Target}) P' \)

from this obtain \( S' \) where \( A1: S \nRightarrow \text{Source S'} \) and \( A2: S' \in S P' \)

by (auto simp add: \text{STCal-step(1)})

from \( A1 \) oc obtain \( T \) where \( A3: [S] \nRightarrow \text{Target T} \) and \( A4: ([S'], T) \in \text{TRel} \)

by \( \text{blast} \)

from \( A3 \) have \( \text{TargetTerm} ([S]) \nRightarrow (\text{STCal Source Target}) (\text{TargetTerm T}) \)

by (simp add: \text{STCal-step(2)})

moreover have \( P' \sim^T [T<\text{TRel}> \text{TargetTerm T} \)

\( \text{proof} \) –
from \( A \) have \( P' \sim [\cdot] T < T \text{Rel} > \) \( \text{TargetTerm} \) \((\text{[S]}\))

by \((\text{simp add: indRelTEQ.ENC})\)

moreover from \( A \) have \( \text{TargetTerm} \) \((\text{[S]}\)) \( ~[\cdot] T < T \text{Rel} > \) \( \text{TargetTerm} \) \( T \)

by \((\text{rule indRelTEQ.target})\)

ultimately show \( P' \sim [\cdot] T < T \text{Rel} > \) \( \text{TargetTerm} \) \( T \)

by \((\text{rule indRelTEQ.trans})\)

qed

ultimately show \( \exists Q'. \) \( \text{TargetTerm} \) \((\text{[S]}\)) \( \rightarrow (\text{STCal Source Target}) \) \( Q' \land P' \sim [\cdot] T < T \text{Rel} > \) \( Q'\)

by blast

next

case \((\text{encL} S)\)

assume \( \text{TargetTerm} \) \((\text{[S]}\)) \( \rightarrow (\text{STCal Source Target}) \) \( P'\)

from this obtain \( T \) where \( B1: \) \((\text{[S]}\)) \( \rightarrow \) \( \text{Target T} \) and \( B2: \) \( T \in T P'\)

by \((\text{auto simp add: STCal-step(2)})\)

from \( B1 \) or obtain \( S' \) where \( B3: \) \( S \rightarrow \) \( \text{Source S} \) and \( B4: \) \((\text{[S']}\), \( T) \in T \text{Rel}\)

by blast

from \( B3 \) have \( \text{SourceTerm} \) \( S \rightarrow (\text{STCal Source Target}) \) \( \text{(SourceTerm S')}\)

by \((\text{simp add: STCal-step(1)})\)

moreover have \( P' \sim [\cdot] T < T \text{Rel} > \) \( \text{SourceTerm} \) \( S'\)

by \((\text{rule indRelTEQ.trans})\)

qed

ultimately show \( \exists Q'. \) \( \text{SourceTerm} \) \( S \rightarrow (\text{STCal Source Target}) \) \( Q' \land P' \sim [\cdot] T < T \text{Rel} > \) \( Q'\)

by blast

next

case \((\text{target T1 T2})\)

assume \( \text{TargetTerm T1} \rightarrow (\text{STCal Source Target}) \) \( P'\)

from this obtain \( T' \) where \( C1: \) \( T1 \rightarrow \) \( \text{Target T'}\) and \( C2: \) \( T1' \in T P'\)

by \((\text{auto simp add: STCal-step(2)})\)

assume \((\text{T1, T2}) \in T \text{Rel}\)

with \( C1 \) bisimT obtain \( T2' \) where \( C3: \) \( T2 \rightarrow \) \( \text{Target T2'}\) and \( C4: \) \((\text{T1', T2'})\) \( \in T \text{Rel}\)

by blast

from \( C3 \) have \( \text{TargetTerm T2} \rightarrow (\text{STCal Source Target}) \) \( \text{(TargetTerm T2')}\)

by \((\text{simp add: STCal-step(2)})\)

moreover from \( C2 C4 \) have \( P' \sim [\cdot] T < T \text{Rel} > \) \( \text{TargetTerm T2'}\)

by \((\text{simp add: indRelTEQ.target})\)

ultimately show \( \exists Q'. \) \( \text{TargetTerm T2} \rightarrow (\text{STCal Source Target}) \) \( Q' \land P' \sim [\cdot] T < T \text{Rel} > \) \( Q'\)

by blast

next

case \((\text{trans P Q R})\)

assume \( P 
\rightarrow (\text{STCal Source Target}) \) \( P'\)

and \( \land P'. \) \( P 
\rightarrow (\text{STCal Source Target}) \) \( P'\)

\[ \Rightarrow \exists Q'. \) \( Q 
\rightarrow (\text{STCal Source Target}) \) \( Q' \land P' \sim [\cdot] T < T \text{Rel} > \) \( Q'\)

from this obtain \( Q' \) where \( D1: \) \( Q 
\rightarrow (\text{STCal Source Target}) \) \( Q' \) and \( D2: \) \( P' \sim [\cdot] T < T \text{Rel} > \) \( Q'\)

by blast

assume \( \land Q'. \) \( Q 
\rightarrow (\text{STCal Source Target}) \) \( Q'\)

\[ \Rightarrow \exists R'. \) \( R 
\rightarrow (\text{STCal Source Target}) \) \( R' \land Q' \sim [\cdot] T < T \text{Rel} > \) \( R'\)

with \( D1 \) obtain \( R' \) where \( D3: \) \( R 
\rightarrow (\text{STCal Source Target}) \) \( R' \) and \( D4: \) \( Q' \sim [\cdot] T < T \text{Rel} > \) \( R'\)

by blast

from \( D2 D4 \) have \( P' \sim [\cdot] T < T \text{Rel} > \) \( R'\)

by \((\text{rule indRelTEQ.trans})\)

with \( D3 \) show \( \exists R'. \) \( R 
\rightarrow (\text{STCal Source Target}) \) \( R' \land P' \sim [\cdot] T < T \text{Rel} > \) \( R'\)

by blast

196
proof (induct arbitrary: Q')

\[
\begin{align*}
\text{case (encR \( S \))} & \quad \text{assume TargetTerm \([S]\) \( \stackrel{\cdot}{\Rightarrow} \text{(STCal Source Target)} \) \( Q' \)} \\
& \quad \text{from this obtain \( T \) where \( E1: [S] \stackrel{\cdot}{\Rightarrow} \text{Target T and E2: } T \in T \) \( Q' \)} \\
& \quad \quad \text{(auto simp add: STCal-step(2))} \\
& \quad \text{from \( E1 \) oc obtain \( S' \) where \( E3: S \Rightarrow \text{Source S' and E4: } ([S], T) \in TRel \)} \\
& \quad \quad \text{by blast} \\
& \quad \text{from \( E3 \) have SourceTerm \( S \Rightarrow \text{(STCal Source Target)} \) (SourceTerm \( S' \))} \\
& \quad \quad \text{by (simp add: STCal-step(1))} \\
& \quad \text{moreover have SourceTerm \( S' \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{by (rule indRelTEQ,encR)} \\
& \quad \text{moreover from \( E2 E4 \) have TargetTerm \([S']\) \( \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{(auto simp add: indRelTEQ,target)} \\
& \quad \text{ultimately show SourceTerm \( S' \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{(by (rule indRelTEQ,trans))} \\
\end{align*}
\]

qed ultimately show \( \exists P'. \text{SourceTerm } [S] \Rightarrow \text{(STCal Source Target)} \) \( P' \land P' \Rightarrow [\cdot] T < TRel > Q' \) \\
\quad by blast

next

\[
\begin{align*}
\text{case (encL \( S \))} & \quad \text{assume SourceTerm \( S \Rightarrow \text{(STCal Source Target)} \) \( Q' \)} \\
& \quad \text{from this obtain \( S' \) where \( F1: S \Rightarrow \text{Source S' and F2: } S' \in S \) \( Q' \)} \\
& \quad \quad \text{(auto simp add: STCal-step(1))} \\
& \quad \text{from \( F1 \) oc obtain \( T \) where \( F3: [S] \Rightarrow \text{Target T and F4: } ([S], T) \in TRel \)} \\
& \quad \quad \text{by blast} \\
& \quad \text{from \( F3 \) have TargetTerm \([S]\) \( \Rightarrow \text{(STCal Source Target)} \) (TargetTerm \( T \))} \\
& \quad \quad \text{(by (simp add: STCal-step(2)))} \\
& \quad \text{moreover have TargetTerm \( T \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{by (rule indRelTEQ,trans)} \\
& \quad \text{ultimately show \( \exists P'. \text{TargetTerm } [S] \Rightarrow \text{(STCal Source Target)} \) \( P' \land P' \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{(by blast)}
\end{align*}
\]

next

\[
\begin{align*}
\text{case (target \( T1 T2 \))} & \quad \text{assume TargetTerm \( T2 \Rightarrow \text{(STCal Source Target)} \) \( Q' \)} \\
& \quad \text{from this obtain \( T2' \) where \( G1: T2 \Rightarrow \text{Target T2' and G2: } T2' \in T \) \( Q' \)} \\
& \quad \quad \text{(auto simp add: STCal-step(2)))} \\
& \quad \text{assume \( (T1, T2) \in TRel \)} \\
& \quad \text{with \( G1 \) bisimT obtain \( T1' \) where \( G3: T1 \Rightarrow \text{Target T1' and G4: } (T1', T2') \in TRel \)} \\
& \quad \quad \text{by blast} \\
& \quad \text{from \( G3 \) have TargetTerm \( T1 \Rightarrow \text{(STCal Source Target)} \) (TargetTerm \( T1' \))} \\
& \quad \quad \text{(by (simp add: STCal-step(2)))} \\
& \quad \text{moreover from \( G2 G4 \) have TargetTerm \( T1' \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{(by (simp add: indRelTEQ,target))} \\
& \quad \text{ultimately show \( \exists P'. \text{TargetTerm } T1 \Rightarrow \text{(STCal Source Target)} \) \( P' \land P' \Rightarrow [\cdot] T < TRel > Q' \)} \\
& \quad \quad \text{(by blast)}
\end{align*}
\]

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

198
with J2 J4 have \( ([S'], T) \in TRel \)
  using indRelTEQ-to-TRel(2)[where TRel=TRel] 
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  by blast
with J3 show \( \exists S'. S \mapsto Source S' \land ([S'], T) \in TRel \)
  by blast
qed
moreover have strong-reduction-bisimulation TRel Target
proof  
  from eqT have TRel" = TRel
  using reflcl-trancl[of TRel] trancl-id[of TRel]
  unfolding equiv-def refl-on-def
  by auto
with bisim show strong-reduction-bisimulation TRel Target
  using indRelTEQ-impl-TRel-is-strong-reduction-bisimulation[where TRel=TRel]
  by simp
qed
ultimately
show strongly-operational-corresponding TRel \land strong-reduction-bisimulation TRel Target
  by simp
qed

lemma (in encoding) SOC-wrt-equivalence-iff-strong-reduction-bisimulation:  
  fixes TRel :: (procT \times 'procT) set
  assumes eqT: equivalence TRel
  shows (strongly-operational-corresponding TRel \land strong-reduction-bisimulation TRel Target) 
    \langle S. (SourceTerm S, TargetTerm ([S])) \in Rel \land (TargetTerm ([S]), SourceTerm S) \in Rel \rangle 
    \land TRel = \{ (T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel \} 
    \land trans Rel \land strong-reduction-bisimulation Rel (STCal Source Target)
  proof (rule iffI, erule conjE)
    assume oc: strongly-operational-corresponding TRel
    and bisimT: strongly-reduction-bisimulation TRel Target
    from eqT have rt: TRel" = TRel
      using reflcl-trancl[of TRel] trancl-id[of TRel]
      unfolding equiv-def refl-on-def
      by auto
    have \forall S. SourceTerm S \sim\[\[ T\langle TRel\rangle\] TargetTerm ([S]) \land TargetTerm ([S]) \sim\[\[ T\langle TRel\rangle\] SourceTerm S
      by (simp add: indRelTEQ.encR indRelTEQ.encL)
    moreover from rt have TRel = \{ (T1, T2), TargetTerm T1 \sim\[\[ T\langle TRel\rangle\] TargetTerm T2 \} 
      using indRelTEQ-to-TRel(4)[where TRel=TRel] 
      trans-closure-of-TRel-refl-cond[where TRel=TRel]
      by (auto simp add: indRelTEQ.target)
    moreover have trans (indRelTEQ TRel)
      using indRelTEQ.trans[where TRel=TRel]
      unfolding trans-def
      by blast
    moreover from eqT oc bisimT
    have strong-reduction-bisimulation (indRelTEQ TRel) (STCal Source Target)
      using SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation[where TRel=TRel]
      by blast
    ultimately
    show \exists Rel. \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \land (TargetTerm ([S]), SourceTerm S) \in Rel 
      \land TRel = \{ (T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel \} \land trans Rel 
      \land strong-reduction-bisimulation Rel (STCal Source Target)
      by blast
next
  assume \exists Rel. \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel 
      \land (TargetTerm ([S]), SourceTerm S) \in Rel 
      \land TRel = \{ (T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel \} \land trans Rel
\[\begin{align*}
&\text{and strongly-operational-corresponding } T_{\text{Rel}} \\
&\text{have } T_{\text{Rel}} \text{ where } A1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in T_{\text{Rel}} \\
&\text{and } A2: T_{\text{Rel}} = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in T_{\text{Rel}}\} \text{ and } A3: \text{trans } T_{\text{Rel}} \\
&\text{by } \text{blast} \\
&\text{obtain } Q' \text{ where } B1: (\text{TargetTerm } ([S]) \rightarrow \text{STCal Source Target}) \rightarrow (\text{STCal Source Target}) Q' \\
&\text{and } B2: (\text{SourceTerm } S', Q') \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: (\text{SourceTerm } S \rightarrow \text{STCal Source Target}) \rightarrow (\text{STCal Source Target}) P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(1)} \\
&\text{ultimately obtain } P' \text{ where } C1: \text{SourceTerm } S \rightarrow \text{STCal Source Target} P' \\
&\text{and } C2: (P', \text{TargetTerm } T) \in T_{\text{Rel}} \\
&\text{by } \text{simp add: STCal-step(2)}
\end{align*}\]
ultimately obtain $Q'$ where $D1: \text{TargetTerm } TQ \mapsto \text{(STCal Source Target) } Q'$
and $D2: (\text{TargetTerm } TP', Q') \in \text{Rel}$

using $A4$
by blast
from $D1$ obtain $TQ'$ where $D3: TQ \mapsto \text{Target } TQ'$ and $D4: TQ' \in T Q'$
by (auto simp add: STCal-step(2))
from $A2 D2 D4$ have $(TP', TQ') \in \text{TRel}$
by simp
with $D3$ show $\exists TQ'. TQ \mapsto \text{Target } TQ' \land (TP', TQ') \in \text{TRel}$
by blast

next
fix $TP$ $TQ$ $TQ'$
assume $(TP, TQ) \in \text{TRel}$
with $A2$ have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$
by simp
moreover assume $TQ \mapsto \text{Target } TQ'$
hence $\text{TargetTerm } TQ \mapsto \text{(STCal Source Target) } (\text{TargetTerm } TQ')$
by (simp add: STCal-step(2))
ultimately obtain $P'$ where $E1: \text{TargetTerm } TP \mapsto \text{(STCal Source Target) } P'$
and $E2: (P', \text{TargetTerm } TQ') \in \text{Rel}$
using $A4$
by blast
from $E1$ obtain $TP'$ where $E3: TP \mapsto \text{Target } TP'$ and $E4: TP' \in T P'$
by (auto simp add: STCal-step(2))
from $A2 E2 E4$ have $(TP', TQ') \in \text{TRel}$
by simp
with $E3$ show $\exists TP'. TP \mapsto \text{Target } TP' \land (TP', TQ') \in \text{TRel}$
by blast
qed
ultimately
show $\text{strongly-operational-corresponding } \text{TRel} \land \text{strong-reduction-bisimulation } \text{TRel } \text{Target}$
by simp
qed

end

theory FullAbstraction
  imports SourceTargetRelation
begin

9 Full Abstraction

An encoding is fully abstract w.r.t. some source term relation $SRel$ and some target term relation $\text{TRel}$ if two source terms $S1$ and $S2$ form a pair $(S1, S2)$ in $SRel$ iff their literal translations form a pair $(\text{enc } S1, \text{enc } S2)$ in $\text{TRel}$.

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

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-in-fully-abstract-wrt-identities:
  assumes injectivity: $\forall S1 S2. [S1] = [S2] \longrightarrow S1 = S2$
  shows fully-abstract $\{(S1, S2), S1 = S2\} \{(T1, T2), T1 = T2\}$
  by (auto simp add: injectivity)

Each encoding is fully abstract w.r.t. the empty relation on the source and the target.
lemma (in encoding) fully-abstract-wrt-empty-relation:
  shows fully-abstract {} {}
  by auto

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

lemma (in encoding) fully-abstract-wrt-all-relation:
  shows fully-abstract {(S1, S2). True} {(T1, T2). True}
  by auto

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

lemma (in encoding) fully-abstract-wrt-source-relation:
  fixes RelS :: ('procS × 'procS) set
  assumes injectivity: ∀ S1 S2. [S1] = [S2] −→ S1 = S2
  shows ∃ RelT. fully-abstract RelS RelT
proof −
  def RelT ≡{(T1, T2). ∃ S1 S2. (S1, S2) ∈ RelS ∧ T1 = [S1] ∧ T2 = [S2]}
  with injectivity have fully-abstract RelS RelT
  by blast
thus ∃ RelT. fully-abstract RelS RelT
  by blast
qed

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

lemma (in encoding) fully-abstract-wrt-trans-source-relation:
  fixes RelS :: ('procS × 'procS) set
  assumes encRelS: ∀ S1 S2. [S1] = [S2] −→ (S1, S2) ∈ RelS
  and transS: trans RelS
  shows ∃ RelT. fully-abstract RelS RelT
proof −
  def relT ≡{(T1, T2). ∃ S1 S2. (S1, S2) ∈ RelS ∧ T1 = [S1] ∧ T2 = [S2]}
  have fully-abstract RelS RelT
  proof auto
  fix S1 S2
  assume (S1, S2) ∈ RelS
  with relT show ([S1], [S2]) ∈ RelT
  by blast
next
  fix S1 S2
  assume ([S1], [S2]) ∈ RelT
  with relT obtain S1′ S2′ where A1: (S1′, S2′) ∈ RelS and A2: [S1] = [S1′] and A3: [S2] = [S2′]
  by blast
from A2 encRelS have (S1, S1′) ∈ RelS
  by simp
from this A1 transS have (S1, S2′) ∈ RelS
  unfolding trans-def
  by blast
moreover from A3 encRelS have (S2′, S2) ∈ RelS
  by simp
ultimately show (S1, S2) ∈ RelS
  using transS
  unfolding trans-def
  by blast
qed
thus ∃ RelT. fully-abstract RelS RelT
  by blast
qed
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
  using encRelS trans-trancl[of RelS]
  fully-abstract-wrt-trans-source-relation[where RelS=RelS⁺]
  by blast

For every encoding and every target term relation RelT there exists a source term relation RelS such that the encoding is fully abstract w.r.t. RelS and RelT.

lemma (in encoding) fully-abstract-wrt-target-relation:

  fixes RelT :: ('procT × 'procT) set
  shows ∃ RelS. fully-abstract RelS RelT

proof –
  def RelS≡{(S1, S2). ([S1], [S2]) ∈ RelT}
  hence fully-abstract RelS RelT
  by simp
  thus ∃ RelS. fully-abstract RelS RelT
  by blast
qed

9.2 Fully Abstract Encodings

Thus, as long as we can choose one of the two relations, full abstraction is trivial. For fixed source and target term relations encodings are not trivially fully abstract. For all encodings and relations SRel and TRel we can construct a relation on the disjunctive union of source and target terms, whose reduction to source terms is SRel and whose reduction to target terms is TRel. But full abstraction ensures that each trans relation that relates source terms and their literal translations in both directions includes SRel if 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: (SourceTerm S, TargetTerm ([S])) ∈ Rel
  and srel: SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
  and trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
  shows ∀ S1 S2. ([S1], [S2]) ∈ TRel ⟷ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel

proof
  auto
  fix S1 S2
  def rel’: Rel’≡Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q}
  hence (TargetTerm ([S1]), SourceTerm S1) ∈ Rel’
  by simp
  moreover assume ([S1], [S2]) ∈ TRel
  with fullAbs have (S1, S2) ∈ SRel
  by simp
  with srel rel’ have (SourceTerm S1, SourceTerm S2) ∈ Rel’
  by simp
  moreover from encR rel’ have (SourceTerm S2, TargetTerm ([S2])) ∈ Rel’
  by simp
  ultimately show (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
  using trans rel’
  unfolding trans-def
  by blast
next
  fix S1 S2
  def rel’: Rel’≡Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q}
  from encR rel’ have (SourceTerm S1, TargetTerm ([S1])) ∈ Rel’
proof

moreover assume (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
with rel' have (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel'
bysimp

moreover from rel' have (TargetTerm ([S2]), SourceTerm S2) ∈ Rel'
bysimp

ultimately have (SourceTerm S1, SourceTerm S2) ∈ Rel
  using trans rel'
  unfolding trans-def
  by blast
with srel have (S1, S2) ∈ SRel
  by ssimp
with fullAbs show ([S1], [S2]) ∈ TRel
  by ssimp
qed

lemma (in encoding) full-abstraction-and-trans-relation-contains-TRel-impl-SRel:

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  and SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
  and encR: ∀S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  and trel: ∀S1 S2. ([S1], [S2]) ∈ TRel ⟷ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
  and trans: trans (Rel ∪ {(P, Q). ∃S. [S] ∈ T P ∧ S ∈ S Q})

shows SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}

proof auto

fix S1 S2
def rel': Rel' ≡ Rel ∪ {(P, Q). ∃S. [S] ∈ T P ∧ S ∈ S Q}
from encR rel' have (SourceTerm S1, TargetTerm ([S1])) ∈ Rel'
bysimp

moreover assume (S1, S2) ∈ SRel
with fullAbs have ([S1], [S2]) ∈ TRel
  by ssimp

with trel rel' have (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel'
bysimp

moreover from rel' have (TargetTerm ([S2]), SourceTerm S2) ∈ Rel'
bysimp

ultimately show (SourceTerm S1, SourceTerm S2) ∈ Rel
  using trans rel'
  unfolding trans-def
  by blast
next

fix S1 S2
def rel': Rel'' ≡ Rel ∪ {(P, Q). ∃S. [S] ∈ T P ∧ S ∈ S Q}
hence (TargetTerm ([S1]), SourceTerm S1) ∈ Rel'
bysimp

moreover assume (SourceTerm S1, SourceTerm S2) ∈ Rel
with rel' have (SourceTerm S1, SourceTerm S2) ∈ Rel'
bysimp

moreover from encR rel' have (SourceTerm S2, TargetTerm ([S2])) ∈ Rel'
bysimp

ultimately have (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
  using trans rel'
  unfolding trans-def
  by blast
with trel have ([S1], [S2]) ∈ TRel
  by ssimp

with fullAbs show (S1, S2) ∈ SRel
  by ssimp
qed
lemma (in encoding) full-abstraction-impl-trans-relation-contains-SRel-iff-TRel:

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
and encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and trans: trans (Rel ∪ {(P, Q). ∃ S. ([S] ∈ T P ∧ S ∈ S Q)})

shows (∀ S1 S2. ([[S1], [S2]]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel)
    <-> (SRel = {([S1], [S2]), (SourceTerm S1, SourceTerm S2) ∈ Rel})
proof
  assume SRel = {([S1], [S2]), (SourceTerm S1, SourceTerm S2) ∈ Rel}
  thus SRel = {([S1], [S2]), (SourceTerm S1, SourceTerm S2) ∈ Rel}
    using assms full-abstraction-and-trans-relation-contains-TRel-impl-SRel
      [where SRel=SRel and TRel=TRel]
    by blast
next
  assume SRel = {([S1], [S2]), (SourceTerm S1, SourceTerm S2) ∈ Rel}
  thus ∀ S1 S2. ([[S1], [S2]]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
    using assms full-abstraction-and-trans-relation-contains-SRel-impl-TRel
      [where SRel=SRel and TRel=TRel]
    by blast
qed

lemma (in encoding) full-abstraction-impl-trans-relation-contains-SRel-iff-TRel-encRL:

fixes Rel :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
and SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
and encR: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and encL: ∀ S. (TargetTerm ([S]), SourceTerm S) ∈ Rel
and trans: trans Rel

shows (∀ S1 S2. ([[S1], [S2]]) ∈ TRel ←→ (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel)
    <-> (SRel = {([S1], [S2]), (SourceTerm S1, SourceTerm S2) ∈ Rel})
proof
  from encL have Rel ∪ {(P, Q). ∃ S. ([S] ∈ T P ∧ S ∈ S Q) = Rel
    by auto
  with fullAbs encR trans show ?thesis
    using full-abstraction-impl-trans-relation-contains-SRel-iff-TRel
      [where Rel=Rel and SRel=SRel and TRel=TRel]
    by simp
qed

Full abstraction ensures that SRel and TRel satisfy the same basic properties that can be defined on
their pairs. In particular: (1) SRel is refl if TRel reduced to translated source terms is refl (2) if the
encoding is surjective then SRel is refl iff TRel is refl (3) SRel is sym iff TRel reduced to translated
source terms is sym (4) SRel is trans iff TRel reduced to translated source terms is trans

lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-refl:

fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel

shows refl SRel <-> (∀ S. ([S], [S]) ∈ TRel)
  unfolding refl-on-def
  by (simp add: fullAbs)

lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-refl:

fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
and surj: ∀ T. ∃ S. T = [S]

shows refl SRel <-> refl TRel
proof
  assume reflS: refl SRel
  show refl TRel
    unfolding refl-on-def
    proof auto
      fix T
      from surj obtain S where T = [S]
      by blast
      moreover from reflS have (S, S) ∈ SRel
        unfolding refl-on-def
        by simp
      with fullAbs have ([S], [S]) ∈ TRel
        by simp
      ultimately show (T, T) ∈ TRel
        by simp
    qed
  next
    assume refl TRel
    with fullAbs show refl SRel
      unfolding refl-on-def
      by simp
    qed

lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-sym:
  fixes SRel :: ('procS * 'procS) set
  and TRel :: ('procT * 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
  shows sym SRel ←→ sym {'(T1, T2). ∃ S1 S2. T1 = [S1] ∧ T2 = [S2] ∧ (T1, T2) ∈ TRel}
    unfolding sym-def
    by (simp add: fullAbs, blast)

lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-sym:
  fixes SRel :: ('procS * 'procS) set
  and TRel :: ('procT * 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
  and surj: ∀ T. ∃ S. T = [S]
  shows sym SRel ←→ sym TRel
    using fullAbs surj
    full-abstraction-impl-SRel-iff-TRel-is-sym[where SRel=SRel and TRel=TRel]
    by auto

lemma (in encoding) full-abstraction-impl-SRel-iff-TRel-is-trans:
  fixes SRel :: ('procS * 'procS) set
  and TRel :: ('procT * 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
  shows trans SRel ←→ trans {'(T1, T2). ∃ S1 S2. T1 = [S1] ∧ T2 = [S2] ∧ (T1, T2) ∈ TRel}
    unfolding trans-def
    by (simp add: fullAbs, blast)

lemma (in encoding) full-abstraction-and-surjectivity-impl-SRel-iff-TRel-is-trans:
  fixes SRel :: ('procS * 'procS) set
  and TRel :: ('procT * 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
  and surj: ∀ T. ∃ S. T = [S]
  shows trans SRel ←→ trans TRel
    using fullAbs surj
    full-abstraction-impl-SRel-iff-TRel-is-trans[where SRel=SRel and TRel=TRel]
    by auto

Similarly, a fully abstract encoding that respects a predicate ensures the this predicate is preserved, reflected, or respected by SRel iff it is preserved, reflected, or respected by TRel.
lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-preserve:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and Pred :: ('procS, 'procT) Proc ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-pred Pred
shows rel-preserves-pred {((P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
↔ rel-preserves-pred {((P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
proof clarify
assume presS: rel-preserves-pred {((P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
show rel-preserves-pred {((P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
proof clarify
fix SP SQ
assume Pred (TargetTerm ([SP]))
with encP have Pred (SourceTerm SP)
  by simp
moreover assume ([SP], [SQ]) ∈ TRel
with fullAbs have (SP, SQ) ∈ SRel
  by simp
ultimately have Pred (SourceTerm SQ)
    using presS
    by blast
with encP show Pred (SourceTerm SQ)
  by simp
qed
next
assume presT:
  rel-preserves-pred {((P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
show rel-preserves-pred {((P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
proof clarify
fix SP SQ
assume Pred (SourceTerm SP)
with encP have Pred (TargetTerm ([SP]))
  by simp
moreover assume (SP, SQ) ∈ SRel
with fullAbs have ([SP], [SQ]) ∈ TRel
  by simp
ultimately have Pred (TargetTerm ([SQ]))
    using presT
    by blast
with encP show Pred (SourceTerm SQ)
  by simp
qed
qed

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-preserve:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-binary-pred Pred
shows rel-preserves-binary-pred {((P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
↔ rel-preserves-binary-pred {((P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
proof clarify
assume presS:
  rel-preserves-binary-pred {((P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
show rel-preserves-binary-pred {((P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
proof clarify
fix x SP SQ
assume \(\text{Pred} (\text{TargetTerm} ([\text{SP}])) \ x\)
with \(\text{encP}\) have \(\text{Pred} (\text{SourceTerm} \ SP) \ x\)
by simp
moreover assume \([\text{SP}], [\text{SQ}]\) \(\in\) \(\text{TRel}\)
with \(\text{fullAbs}\) have \((\text{SP}, \text{SQ})\) \(\in\) \(\text{SRel}\)
by simp
ultimately have \(\text{Pred} (\text{SourceTerm} \ SQ) \ x\)
using \(\text{presS}\)
by blast
with \(\text{encP}\) show \(\text{Pred} (\text{TargetTerm} ([\text{SQ}])) \ x\)
by simp
qed

next
assume \(\text{presT}\): rel-preserves-binary-pred \(\{(P, Q), \exists SP SQ. [SP] \in\ T P \land [SQ] \in\ T Q \land ([SP], [SQ]) \in \text{TRel}\}\) \(\text{Pred}\)
show rel-preserves-binary-pred \(\{(P, Q), \exists SP SQ. SP \in\ S P \land SQ \in\ S Q \land (SP, SQ) \in \text{SRel}\}\) \(\text{Pred}\)
proof clarify
fix \(x SP SQ\)
assume \(\text{Pred} (\text{SourceTerm} \ SP) \ x\)
with \(\text{encP}\) have \(\text{Pred} (\text{TargetTerm} ([\text{SP}])) \ x\)
by simp
moreover assume \((\text{SP}, \text{SQ})\) \(\in\) \(\text{SRel}\)
with \(\text{fullAbs}\) have \((\text{SP}, [\text{SQ}])\) \(\in\) \(\text{TRel}\)
by simp
ultimately have \(\text{Pred} (\text{TargetTerm} ([\text{SQ}])) \ x\)
using \(\text{presT}\)
by blast
with \(\text{encP}\) show \(\text{Pred} (\text{SourceTerm} \ SQ) \ x\)
by simp
qed

next

lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects:
fixes \(\text{SRel}:: (\text{procS} \times \text{procS})\) \(\text{set}\)
and \(\text{TRel}:: (\text{procT} \times \text{procT})\) \(\text{set}\)
and \(\text{Pred}:: (\text{procS}, \text{procT})\) \(\text{Proc} \Rightarrow \text{bool}\)
assumes \(\text{fullAbs}:: \text{fully-abstract} \text{SRel} \text{TRel}\)
and \(\text{encP}:: \text{enc-respects-pred} \text{Pred}\)
shows rel-reflects-pred \(\{(P, Q), \exists SP SQ. SP \in\ S P \land SQ \in\ S Q \land (SP, SQ) \in \text{SRel}\}\) \(\text{Pred}\)
\(\iff\) rel-reflects-pred \(\{(P, Q), \exists SP SQ. [SP] \in\ T P \land [SQ] \in\ T Q \land ([SP], [SQ]) \in \text{TRel}\}\) \(\text{Pred}\)
proof clarify
fix \(SP SQ\)
assume \(\text{Pred} (\text{TargetTerm} ([\text{SP}]))\)
with \(\text{encP}\) have \(\text{Pred} (\text{SourceTerm} \ SQ)\)
by simp
moreover assume \([\text{SP}], [\text{SQ}]\) \(\in\) \(\text{TRel}\)
with \(\text{fullAbs}\) have \((\text{SP}, [\text{SQ}])\) \(\in\) \(\text{SRel}\)
by simp
ultimately have \(\text{Pred} (\text{SourceTerm} \ SP)\)
using \(\text{refS}\)
by blast
with \(\text{encP}\) show \(\text{Pred} (\text{TargetTerm} ([\text{SP}]))\)
by simp
qed

next
assume \(\text{refT}\):
rel-reflects-pred \(\{(P, Q), \exists SP SQ. [SP] \in\ T P \land [SQ] \in\ T Q \land ([SP], [SQ]) \in \text{TRel}\}\) \(\text{Pred}\)
show rel-reflects-pred \(\{(P, Q), \exists SP SQ. SP \in\ S P \land SQ \in\ S Q \land (SP, SQ) \in \text{SRel}\}\) \(\text{Pred}\)
proof clarify
fix SP SQ
assume Pred (SourceTerm SQ)
with encP have Pred (TargetTerm ([SQ]))
by simp
moreover assume (SP, SQ) ∈ SRel
with fullAbs have ([SP], [SQ]) ∈ TRel
by simp
ultimately have Pred (TargetTerm ([SP]))
    using reflT
    by blast
with encP show Pred (SourceTerm SP)
    by simp
qed
qed

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
and Pred :: ('procS, 'procT) Proc ⇒ 'b ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-binary-pred Pred
shows rel-reflects-binary-pred \{(P, Q). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel\} Pred
    ⇔ rel-reflects-binary-pred \{(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel\} Pred
proof clarify
fix x SP SQ
assume Pred (TargetTerm ([SQ])) x
with encP have Pred (SourceTerm SQ) x
by simp
moreover assume ([SP], [SQ]) ∈ TRel
with fullAbs have (SP, SQ) ∈ SRel
by simp
ultimately have Pred (SourceTerm SP) x
    using reflS
    by blast
with encP show Pred (TargetTerm ([SP])) x
    by simp
qed
next
assume reflT:
rel-reflects-binary-pred \{(P, Q). \exists SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel\} Pred
show rel-reflects-binary-pred \{(P, Q). \exists SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel\} Pred
proof clarify
fix x SP SQ
assume Pred (SourceTerm SQ) x
with encP have Pred (TargetTerm ([SQ])) x
by simp
moreover assume (SP, SQ) ∈ SRel
with fullAbs have ([SP], [SQ]) ∈ TRel
by simp
ultimately have Pred (TargetTerm ([SP])) x
    using reflT
    by blast
with encP show Pred (SourceTerm SP) x
    by simp
lemma (in encoding) full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-respects:
fixes SRel :: (‘procS × ‘procS) set
and TRel :: (‘procT × ‘procT) set
and Pred :: (‘procS, ‘procT) Proc ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-pred Pred
shows rel-respects-pred {(P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
←→ rel-respects-pred {(P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
using assms full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-preserve[where
SRel=SRel and TRel=TRel and Pred=Pred]
full-abstraction-and-enc-respects-pred-impl-SRel-iff-TRel-reflects[where
SRel=SRel and TRel=TRel and Pred=Pred]
by auto

lemma (in encoding) full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-respects:
fixes SRel :: (‘procS × ‘procS) set
and TRel :: (‘procT × ‘procT) set
and Pred :: (‘procS, ‘procT) Proc ⇒ ‘b ⇒ bool
assumes fullAbs: fully-abstract SRel TRel
and encP: enc-respects-binary-pred Pred
shows rel-respects-binary-pred {(P, Q). ∃ SP SQ. SP ∈ S P ∧ SQ ∈ S Q ∧ (SP, SQ) ∈ SRel} Pred
←→ rel-respects-binary-pred
{(P, Q). ∃ SP SQ. [SP] ∈ T P ∧ [SQ] ∈ T Q ∧ ([SP], [SQ]) ∈ TRel} Pred
using assms full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-preserve[where
SRel=SRel and TRel=TRel and Pred=Pred]
full-abstraction-and-enc-respects-binary-pred-impl-SRel-iff-TRel-reflects[where
SRel=SRel and TRel=TRel and Pred=Pred]
by auto

9.3 Full Abstraction w.r.t. Preorders

If there however exists a trans relation Rel that relates source terms and their literal translations in both directions, then the encoding is fully abstract with respect to the reduction of Rel to source terms and the reduction of Rel to target terms.

lemma (in encoding) trans-source-target-relation-impl-full-abstraction:
assumes enc: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
∧ (TargetTerm ([S]), SourceTerm S) ∈ Rel
and trans: trans Rel
shows fully-abstract {(S1, S2), (SourceTerm S1, SourceTerm S2) ∈ Rel}
{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}
proof auto
fix S1 S2
assume (SourceTerm S1, SourceTerm S2) ∈ Rel
with enc trans show (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
  unfolding trans-def
  by blast
next
fix S1 S2
assume (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
with enc trans show (SourceTerm S1, SourceTerm S2) ∈ Rel
  unfolding trans-def
  by blast
qed

lemma (in encoding) source-target-relation-impl-full-abstraction-wrt-trans-closures:

proof auto
fix $S_1$ $S_2$
from enc have $(\text{TargetTerm } ([S_1])), \text{SourceTerm } S_1) \in \text{Rel}^+$
  by blast
moreover assume $(\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}^+$
ultimately have $(\text{TargetTerm } ([S_1]), \text{SourceTerm } S_2) \in \text{Rel}^+$
  by simp
moreover from enc have $(\text{SourceTerm } S_2, \text{TargetTerm } ([S_2])) \in \text{Rel}^+$
  by blast
ultimately show $(\text{TargetTerm } ([S_1]), \text{TargetTerm } ([S_2])) \in \text{Rel}^+$
  by simp

next
fix $S_1$ $S_2$
from enc have $(\text{SourceTerm } S_1, \text{TargetTerm } ([S_1])) \in \text{Rel}^+$
  by blast
moreover assume $(\text{TargetTerm } ([S_1]), \text{TargetTerm } ([S_2])) \in \text{Rel}^+$
ultimately have $(\text{SourceTerm } S_1, \text{TargetTerm } ([S_2])) \in \text{Rel}^+$
  by simp
moreover from enc have $(\text{TargetTerm } ([S_2]), \text{SourceTerm } S_2) \in \text{Rel}^+$
  by blast
ultimately show $(\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}^+$
  by simp

qed

lemma (in encoding) quasi-trans-source-target-relation-impl-full-abstraction:
  fixes Rel :: (\"\text{\'procS}, \text{\'procT}\text{\'} Proc \times \text{\'procS}, \text{\'procT}\text{\'} Proc\) set
  and SRel :: (\text{\'procS} \times \text{\'procS} set
  and TRel :: (\text{\'procT} \times \text{\'procT} set
  assumes enc: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}$
  $\wedge (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{Rel}$
  and srel: $SRel = \{(S_1, S_2). (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}\}$
  and trel: $TRel = \{(T_1, T_2). (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\}$
  and trans: $\forall P Q R. (P, Q) \in \text{Rel} \wedge (Q, R) \in \text{Rel} \wedge (P \in \text{ProcS} \wedge Q \in \text{ProcT})$
  $\wedge (P \in \text{ProcT} \wedge Q \in \text{ProcS}) \rightarrow (P, R) \in \text{Rel}$
  shows fully-abstract $SRel \rightarrow TRel$
proof auto
fix $S_1$ $S_2$
from enc have $(\text{TargetTerm } ([S_1])), \text{SourceTerm } S_1) \in \text{Rel}$
  by simp
moreover assume $(S_1, S_2) \in SRel$
with srel have $(\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}$
  by simp
ultimately have $(\text{TargetTerm } ([S_1]), \text{SourceTerm } S_2) \in \text{Rel}$
  using trans
  by blast
moreover from enc have $(\text{SourceTerm } S_2, \text{TargetTerm } ([S_2])) \in \text{Rel}$
  by simp
ultimately have $(\text{TargetTerm } ([S_1]), \text{TargetTerm } ([S_2])) \in \text{Rel}$
  using trans
  by blast
with trel show $([S_1], [S_2]) \in TRel$
  by simp
next
fix $S_1$ $S_2$
from enc have $(\text{SourceTerm } S_1, \text{TargetTerm } ([S_1])) \in \text{Rel}$
  by simp
moreover assume $([S_1], [S_2]) \in TRel$
with \( trel \) have \((\text{TargetTerm} ([S1]), \text{TargetTerm} ([S2])) \in \text{Rel}\)
by simp
ultimately have \((\text{SourceTerm} S1, \text{TargetTerm} ([S2])) \in \text{Rel}\)
using trans
by blast
moreover from \( \text{enc} \) have \((\text{TargetTerm} ([S2]), \text{SourceTerm} S2) \in \text{Rel}\)
by simp
ultimately have \((\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\)
using trans
by blast
with \( srel \) show \((S1, S2) \in SRel\)
by simp
qed

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

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

\[
\begin{align*}
\text{fixes} \quad & SRel :: (\text{\texttt{\textquoteleft\text{\textquoteleft}procS \times \\texttt{\textquoteleft\textquoteleft}procS}}} \text{\textquoteleft\textquoteleft}) \text{set} \\
& TRel :: (\text{\texttt{\textquoteleft\textquoteleft}procT \times \\texttt{\textquoteleft\textquoteleft}procT}}) \text{set} \\
& P Q :: (\text{\texttt{\textquoteleft\textquoteleft}procS, \text{\textquoteleft\textquoteleft}procT}} \text{Proc} \\
\text{assumes} \quad & \text{fullAbs: fully-abstract \( SRel \) \( TRel \)} \\
& \text{rel: } P <\leq R <\leq SRel, TRel > Q \\
\text{shows} \quad & \forall SP SQ. \ SP \in S P \land SQ \in S Q \rightarrow ([SP], [SQ]) \in TRel^+ \\
& \forall SP TQ. \ SP \in S P \land TQ \in T Q \rightarrow ([SP], [TQ]) \in TRel^+
\end{align*}
\]

**proof**

\[
\begin{align*}
\text{have fullAbsT: } \forall S1 S2. \ (S1, S2) \in SRel^+ \rightarrow ([S1], [S2]) \in TRel^+
\end{align*}
\]

**proof clarify**

\[
\begin{align*}
\text{fix } S1 S2 \\
\text{assume } (S1, S2) \in SRel^+ \\
\text{thus } ([S1], [S2]) \in TRel^+
\end{align*}
\]

**proof induct**

\[
\begin{align*}
\text{fix } S2 \\
\text{assume } (S1, S2) \in SRel \\
\text{with fullAbs show } ([S1], [S2]) \in TRel^+
\end{align*}
\]

by simp

**next**

\[
\begin{align*}
\text{case } (\text{step } S2 S3) \\
\text{assume } ([S1], [S2]) \in TRel^+ \\
\text{moreover assume } (S2, S3) \in SRel \\
\text{with fullAbs have } ([S2], [S3]) \in TRel^+
\end{align*}
\]

by simp

ultimately show \((S1), [S3]) \in TRel^+

by simp

qed

**qed**

with \( rel \) show \( \forall SP SQ. \ SP \in S P \land SQ \in S Q \rightarrow ([SP], [SQ]) \in TRel^+ \)
using \( \text{indRelRSTPO-to-SRel-and-TRel(1)} \)\( \text{where } SRel=SRel \) \( \text{and } TRel=TRel \)
by simp

show \( \forall SP TQ. \ SP \in S P \land TQ \in T Q \rightarrow ([SP], [TQ]) \in TRel^+ \)

**proof clarify**

\[
\begin{align*}
\text{fix } SP TQ \\
\text{assume } SP \in S P \land TQ \in T Q \\
\text{with } rel \text{ obtain } S \text{ where } A1: \ (SP, S) \in SRel^+ \\
& \text{and } A2: \ ([S], TQ) \in TRel^+
\end{align*}
\]

using \( \text{indRelRSTPO-to-SRel-and-TRel(2)} \)\( \text{where } SRel=SRel \) \( \text{and } TRel=TRel \)
by blast

from \( A1 \) have \( SP = S \lor (SP, S) \in SRel^+ \)
using \( \text{rtrancl-eq-or-trancl\{of } SP S SRel \)
by blast

with \( fullAbsT \) have \((SP), [S]) \in TRel^+ \)

\[212\]
by fast
from this A2 show ([SP], TQ) ∈ TRel
by simp
qed

lemma (in encoding) full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  and P Q :: ('procS, 'procT) Proc
  assumes fA: fully-abstract SRel TRel
  and transT: trans TRel
  and reflS: refl SRel
  and rel: P ~ [[<SRel, TRel> Q
  shows ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([SP], [SQ]) ∈ SRel
  and ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([SP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([TP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → ([TP], [SQ]) ∈ TRel
by simp+
  from reflS fA show ∀ SP TQ. SP ∈ S SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → ([SP], TQ) ∈ TRel
  unfolding refl-on-def
by simp

next
case (encR S)
  show ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([SP], [SQ]) ∈ SRel
  and ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([SP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([TP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → ([TP], [SQ]) ∈ TRel
by simp+
  with reflS fA show ∀ TP SQ. TP ∈ T TargetTerm ([S]) ∧ SQ ∈ S SourceTerm S → ([TP], [SQ]) ∈ TRel
  unfolding refl-on-def
by simp

next
case (encL S)
  show ∀ SP SQ. SP ∈ S SourceTerm S ∧ SQ ∈ S TargetTerm ([S]) → ([SP], [SQ]) ∈ SRel
  and ∀ SP SQ. SP ∈ S TargetTerm ([S]) ∧ SQ ∈ S SourceTerm S → ([SP], [SQ]) ∈ TRel
  and ∀ SP TQ. SP ∈ S TargetTerm ([S]) ∧ TQ ∈ T SourceTerm S → ([TP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm S ∧ TQ ∈ T TargetTerm ([S]) → ([TP], [SQ]) ∈ TRel
by simp+
  assume (S1, S2) ∈ SRel
  thus ∀ SP SQ. SP ∈ S SourceTerm S1 ∧ SQ ∈ S SourceTerm S2 → ([SP], [SQ]) ∈ SRel
by simp
  with fA show ∀ SP SQ. SP ∈ S SourceTerm S1 ∧ SQ ∈ S SourceTerm S2 → ([SP], [SQ]) ∈ TRel
by simp

next
case (target T1 T2)
  show ∀ SP SQ. SP ∈ S SourceTerm T1 ∧ SQ ∈ S SourceTerm T2 → ([SP], [SQ]) ∈ SRel
  and ∀ SP SQ. SP ∈ S SourceTerm T1 ∧ SQ ∈ S SourceTerm T2 → ([SP], [SQ]) ∈ TRel
  and ∀ SP TQ. SP ∈ S SourceTerm T1 ∧ TQ ∈ T SourceTerm T2 → ([TP], [SQ]) ∈ TRel
  and ∀ TP SQ. TP ∈ T SourceTerm T1 ∧ SQ ∈ S SourceTerm T2 → ([TP], [SQ]) ∈ TRel
by simp+
  assume (T1, T2) ∈ TRel
  thus ∀ TP TQ. TP ∈ T SourceTerm T1 ∧ TQ ∈ T SourceTerm T2 → ([TP], [SQ]) ∈ TRel
by simp

213
next

    case (trans P Q R)
    assume A1: ∀ SP SQ. SP ∈ S P ∧ SQ ∈ S Q → ([SP], [SQ]) ∈ TRel
        and A2: ∀ SP TQ. SP ∈ S P ∧ TQ ∈ T Q → ([SP], TQ) ∈ TRel
        and A3: ∀ TP SQ. TP ∈ T P ∧ SQ ∈ S Q → (TP, [SQ]) ∈ TRel
        and A4: ∀ TP TQ. TP ∈ T P ∧ TQ ∈ T Q → (TP, TQ) ∈ TRel
        and A5: ∀ SQ SR. SQ ∈ S Q ∧ SR ∈ S R → ([SQ], [SR]) ∈ TRel
        and A6: ∀ SQ TR. SQ ∈ S Q ∧ TR ∈ T R → ([SQ], TR) ∈ TRel
        and A7: ∀ TQ SR. TQ ∈ T Q ∧ SR ∈ S R → (TQ, [SR]) ∈ TRel
        and A8: ∀ TQ TR. TQ ∈ T Q ∧ TR ∈ T R → (TQ, TR) ∈ TRel
    show ∀ SP SR. SP ∈ S P ∧ SR ∈ S R → ([SP], [SR]) ∈ TRel

show clarify

proof clarify

    fix SP SR
    assume A9: SP ∈ S P and A10: SR ∈ S R
    show ([SP], [SR]) ∈ TRel

proof (cases Q)

    case (SourceTerm SQ)
    assume A11: SQ ∈ S Q
        with A1 A9 have ([SP], [SQ]) ∈ TRel
            by blast
        moreover from A5 A10 A11 have ([SQ], [SR]) ∈ TRel
            by blast
        ultimately show ([SP], [SR]) ∈ TRel
            using transT
            unfolding trans-def
            by blast

next

    case (TargetTerm TQ)
    assume A11: TQ ∈ T Q
        with A2 A9 have ([SP], TQ) ∈ TRel
            by blast
        moreover from A7 A10 A11 have (TQ, [SR]) ∈ TRel
            by blast
        ultimately show ([SP], [SR]) ∈ TRel
            using transT
            unfolding trans-def
            by blast

qed

qed

with JA show ∀ SP SR. SP ∈ S P ∧ SR ∈ S R → (SP, SR) ∈ SRel
    by simp
    show ∀ SP TR. SP ∈ S P ∧ TR ∈ T R → ([SP], TR) ∈ TRel

proof clarify

fix SP TR

    assume A9: SP ∈ S P and A10: TR ∈ T R
    show ([SP], TR) ∈ TRel

proof (cases Q)

    case (SourceTerm SQ)
    assume A11: SQ ∈ S Q
        with A1 A9 have ([SP], [SQ]) ∈ TRel
            by blast
        moreover from A6 A10 A11 have ([SQ], TR) ∈ TRel
            by blast
        ultimately show ([SP], TR) ∈ TRel
            using transT
            unfolding trans-def
            by blast

next

    case (TargetTerm TQ)
    assume A11: TQ ∈ T Q
        with A2 A9 have ([SP], TQ) ∈ TRel
by blast
moreover from A8 A10 A11 have \((TQ, TR) \in TRel\)
by blast
ultimately show \(([SP], TR) \in TRel\)
  using trans\(T\)
  unfolding trans-def
by blast
qed
qed
show \(\forall TP SR. TP \in T P \land SR \in S R \rightarrow (TP, [SR]) \in TRel\)
proof clarify
fix TP SR
assume A9: TP \(\in T P\) and A10: SR \(\in S R\)
show \((TP, [SR]) \in TRel\)
proof (cases Q)
case (SourceTerm SQ)
assume A11: SQ \(\in S Q\)
with A3 A9 have \((TP, [SQ]) \in TRel\)
by blast
moreover from A5 A10 A11 have \([SQ, [SR]) \in TRel\)
by blast
ultimately show \((TP, [SR]) \in TRel\)
  using trans\(T\)
  unfolding trans-def
by blast
next
case (TargetTerm TQ)
assume A11: TQ \(\in T Q\)
with A4 A9 have \((TP, TQ) \in TRel\)
by blast
moreover from A7 A10 A11 have \((TQ, [SR]) \in TRel\)
by blast
ultimately show \((TP, [SR]) \in TRel\)
  using trans\(T\)
  unfolding trans-def
by blast
qed
qed
show \(\forall TP TR. TP \in T P \land TR \in T R \rightarrow (TP, TR) \in TRel\)
proof clarify
fix TP TR
assume A9: TP \(\in T P\) and A10: TR \(\in T R\)
show \((TP, TR) \in TRel\)
proof (cases Q)
case (SourceTerm SQ)
assume A11: SQ \(\in S Q\)
with A3 A9 have \((TP, [SQ]) \in TRel\)
by blast
moreover from A6 A10 A11 have \([SQ, TR) \in TRel\)
by blast
ultimately show \((TP, TR) \in TRel\)
  using trans\(T\)
  unfolding trans-def
by blast
next
case (TargetTerm TQ)
assume A11: TQ \(\in T Q\)
with A4 A9 have \((TP, TQ) \in TRel\)
by blast
moreover from A8 A10 A11 have \((TQ, TR) \in TRel\)
by blast
ultimately show \((TP, TR) \in TRel\)
using \(\text{trans}T\)
unfolding \text{trans-def} 
by \text{blast} 

qed 

qed 

qed

If an encoding is fully abstract w.r.t. a preorder \(SRel\) on the source and a trans relation \(TRel\) on the target, then there exists a trans relation, namely \(\text{indRelSTEQ}\), 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 \(TRel\).

\textbf{lemma (in encoding) full-abstraction-wrt-preorders-impl-trans-source-target-relation:}
fixes \(SRel\) :: (‘procS × ‘procS) set 
and \(TRel\) :: (‘procT × ‘procT) set 
assumes \(\text{fullAbs}: \text{fully-abstract } SRel \ TRel\)
and \(\text{reflS}: \text{refl } SRel\)
and \(\text{transT}: \text{trans } TRel\)
shows \(\exists \text{Rel}. \forall S. \text{SourceTerm } S \sim[S]<SRel,TRel> \text{TargetTerm } ([S]) \in \text{Rel} \land \text{TargetTerm } ([S], \text{SourceTerm } S) \in \text{Rel}\)
\(\land SRel = \{(S1, S2). \text{SourceTerm } S1, \text{SourceTerm } S2) \in \text{Rel}\}\)
\(\land TRel = \{(T1, T2). \text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}\)
\(\land \text{trans } \text{Rel}\)

proof –

have \(\forall S. \text{SourceTerm } S \sim[S]<SRel,TRel> \text{TargetTerm } ([S])\)
\(\land \text{TargetTerm } ([S]) \sim[S]<SRel,TRel> \text{SourceTerm } S\)
using \(\text{indRelSTEQ}.\text{encR}[\text{where } SRel=SRel \text{ and } TRel=TRel]\)
\text{indRelSTEQ}.\text{encL}[\text{where } SRel=SRel \text{ and } TRel=TRel] 

by \text{blast} 

moreover have \(SRel = \{(S1, S2). \text{SourceTerm } S1 \sim[S]<SRel,TRel> \text{SourceTerm } S2\}\)

proof auto

fix \(S1, S2\)
assume \((S1, S2) \in SRel\)
thus \(\text{SourceTerm } S1 \sim[S]<SRel,TRel> \text{SourceTerm } S2\)

by (rule \text{indRelSTEQ}.\text{source}[\text{where } SRel=SRel \text{ and } TRel=TRel])

next

fix \(S1, S2\)
assume \(\text{SourceTerm } S1 \sim[S]<SRel,TRel> \text{SourceTerm } S2\)

with \(\text{fullAbs } \text{reflS } \text{trans}T\) show \((S1, S2) \in SRel\)

using \text{full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel}(1)[\text{where } SRel=SRel \text{ and } TRel=TRel] 

by \text{blast} 

qed

moreover have \(TRel = \{(T1, T2). \text{TargetTerm } T1 \sim[T]<SRel,TRel> \text{TargetTerm } T2\}\)

proof auto

fix \(T1, T2\)
assume \((T1, T2) \in TRel\)
thus \(\text{TargetTerm } T1 \sim[T]<SRel,TRel> \text{TargetTerm } T2\)

by (rule \text{indRelSTEQ}.\text{target}[\text{where } SRel=SRel \text{ and } TRel=TRel])

next

fix \(T1, T2\)
assume \(\text{TargetTerm } T1 \sim[T]<SRel,TRel> \text{TargetTerm } T2\)

with \(\text{fullAbs } \text{reflS } \text{trans}T\) show \((T1, T2) \in TRel\)

using \text{full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel}(5)[\text{where } SRel=SRel \text{ and } TRel=TRel] 

by \text{blast} 

qed

moreover have \(\text{trans } (\text{indRelSTEQ } SRel \ TRel)\)

using \(\text{indRelSTEQ}.\text{trans}[\text{where } SRel=SRel \text{ and } TRel=TRel] 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def} 

unfolding \text{trans-def}
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** $SRel := ('procS \times 'procS)$ set

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

**Shows** $(\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel 
\land (TargetTerm ([S]), SourceTerm S) \in Rel) 
\land SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\} 
\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} 
\land trans Rel) $

**Proof (rule iffI)**

**Assume** $\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel 
\land (TargetTerm ([S]), SourceTerm S) \in Rel) 
\land SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\} 
\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} 
\land trans Rel$ 

**Using** $\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel 
\land (TargetTerm ([S]), SourceTerm S) \in Rel) 
\land SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\} 
\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} 
\land trans Rel$ 

**Fix** $S$

**From** $A1$ have $(SourceTerm S, TargetTerm ([S])) \in Rel$

**By** blast

**Moreover from** $A1$ have $(TargetTerm ([S]), SourceTerm S) \in Rel$

**By** blast

**Ultimately have** $(SourceTerm S, SourceTerm S) \in Rel$

**Using** $A4$

**Unfolding** $trans-def$

**By** blast

**With** $A2$ show $(S, S) \in SRel$

**By** blast

**Qed**

**Moreover from** $A3 A4$ have $TRel$

**Unfolding** $trans-def$

**By** blast

**Ultimately show** $\exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel 
\land (TargetTerm ([S]), SourceTerm S) \in Rel) 
\land SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) \in Rel\} 
\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\} 
\land trans Rel$ 

**By** blast
9.4 Full Abstraction w.r.t. Equivalences

If there exists a relation Rel that relates source terms and their literal translations and whose sym
closure is trans, then the encoding is fully abstract with respect to the reduction of the sym closure
of Rel to source/target terms.

**lemma (in encoding) source-target-relation-with-trans-symcl-impl-full-abstraction:**

**fixes** Rel :: ((‘procS’, ‘procT’) Proc × (‘procS’, ‘procT’) Proc) set

**assumes** enc: ∀ S. (SourceTerm S, TargetTerm ([]S)) ∈ Rel

and trans: trans (symcl Rel)

**shows** fully-abstract {S1, S2}. (SourceTerm S1, SourceTerm S2) ∈ symcl Rel

proof auto

fix S1 S2

from enc have (TargetTerm ([]S1), SourceTerm S1) ∈ symcl Rel

by (simp add: symcl-def)

moreover assume (SourceTerm S1, SourceTerm S2) ∈ symcl Rel

moreover from enc have (SourceTerm S2, TargetTerm ([]S2)) ∈ symcl Rel

by (simp add: symcl-def)

ultimately show (TargetTerm ([]S1), TargetTerm ([]S2)) ∈ symcl Rel

using trans

unfolding trans-def

by blast

next

fix S1 S2

from enc have (SourceTerm S1, TargetTerm ([]S1)) ∈ symcl Rel

by (simp add: symcl-def)

moreover assume (TargetTerm ([]S1), TargetTerm ([]S2)) ∈ symcl Rel

moreover from enc have (TargetTerm ([]S2), SourceTerm S2) ∈ symcl Rel

by (simp add: symcl-def)

ultimately show (SourceTerm S1, SourceTerm S2) ∈ symcl Rel

using trans

unfolding trans-def

by blast

qed

If an encoding is fully abstract w.r.t. the equivalences SRel and TRel, then there exists a preorder,

namely indRelRSTPO, that relates source terms and their literal translations such that its reductions
to source terms is SRel and its reduction to target terms is TRel.

**lemma (in encoding) fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder:**

**fixes** SRel :: (‘procS × ‘procS) set

and TRel :: (‘procT × ‘procT) set

**assumes** fullAbs: fully-abstract SRel TRel

and reflT: refl TRel

and symmT: sym TRel

and transT: trans TRel

**shows** ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([]S)) ∈ Rel)

∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}

∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}

∧ preorder (symcl Rel)

proof –

from fullAbs reflT have reflS: refl SRel

unfolding refl-on-def

by auto

from fullAbs symmT have symmS: sym SRel

unfolding sym-def

by auto

from fullAbs transT have transS: trans SRel

qed
proof  auto
next  auto  proof  refl
moreover  have  qed
by  blast

moreover  have  $S_{\text{Rel}} = \{(S_1, S_2). (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})\}$
proof  auto
fix  $S_1 S_2$
assume  $(S_1, S_2) \in S_{\text{Rel}}$
thus  $(\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})$
by  $(\text{simp add: symcl-def indRelRSTPO.source}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}])$

next
fix  $S_1 S_2$
assume  $(\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})$
moreover from  $\text{transS}$
have  $\text{SourceTerm } S_1 \preceq [1]R<\text{S_{Rel}, T_{Rel}}> \text{SourceTerm } S_2 \implies (S_1, S_2) \in S_{\text{Rel}}$
  using  $\text{indRelRSTPO-to-SRel-and-TRel(1)}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}]$  
  $\text{transcl-id}[\text{of } S_{\text{Rel}}]$
by  blast
moreover from  $\text{symmsS } \text{ transS}$
have  $\text{SourceTerm } S_2 \preceq [1]R<\text{S_{Rel}, T_{Rel}}> \text{SourceTerm } S_1 \implies (S_1, S_2) \in S_{\text{Rel}}$
  using  $\text{indRelRSTPO-to-SRel-and-TRel(1)}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}]$  
  $\text{transcl-id}[\text{of } S_{\text{Rel}}]$
unfolding  $\text{sym-def}$
by  blast
ultimately show  $(S_1, S_2) \in S_{\text{Rel}}$
by  $(\text{auto simp add: symcl-def})$

qed
moreover
have  $T_{\text{Rel}} = \{(T_1, T_2). (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})\}$
proof  auto
fix  $T_1 T_2$
assume  $(T_1, T_2) \in T_{\text{Rel}}$
thus  $(\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})$
by  $(\text{simp add: symcl-def indRelRSTPO.target}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}])$

next
fix  $T_1 T_2$
assume  $(\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})$
moreover from  $\text{transT}$
have  $\text{TargetTerm } T_1 \preceq [1]R<\text{S_{Rel}, T_{Rel}}> \text{TargetTerm } T_2 \implies (T_1, T_2) \in T_{\text{Rel}}$
  using  $\text{indRelRSTPO-to-SRel-and-TRel(4)}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}]$  
  $\text{transcl-id}[\text{of } T_{\text{Rel}}]$
by  blast
moreover from  $\text{symmsT } \text{ transT}$
have  $\text{TargetTerm } T_2 \preceq [1]R<\text{S_{Rel}, T_{Rel}}> \text{TargetTerm } T_1 \implies (T_1, T_2) \in T_{\text{Rel}}$
  using  $\text{indRelRSTPO-to-SRel-and-TRel(4)}[\text{where } S_{\text{Rel}}=S_{\text{Rel}} \text{ and } T_{\text{Rel}}=T_{\text{Rel}}]$  
  $\text{transcl-id}[\text{of } T_{\text{Rel}}]$
unfolding  $\text{sym-def}$
by  blast
ultimately show  $(T_1, T_2) \in T_{\text{Rel}}$
by  $(\text{auto simp add: symcl-def})$

qed
moreover have  refl  $(\text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}}))$
unfolding  refl-on-def
proof  auto
fix  $P$
show  $(P, P) \in \text{symcl } \text{(indRelRSTPO } S_{\text{Rel}} \text{ } T_{\text{Rel}})$
proof  $(\text{cases } P)$
case  $(\text{SourceTerm } SP)$
assume  $SP \in S \text{ } P$

219
proof
clarify

\begin{align*}
\text{assume } & P \land R, P \leq [\cdot R < SRel, TRel], Q \land R \leq [\cdot R < SRel, TRel], Q \land (P, R) \notin (\text{indRelRSTPO} SRel \text{ TRel}) \\
\rightarrow & Q \leq [\cdot R < SRel, TRel] \land P \lor Q \leq [\cdot R < SRel, TRel], R
\end{align*}

proof
clarify

\begin{align*}
\text{assume } & A1: P \leq [\cdot R < SRel, TRel], Q \land R \leq [\cdot R < SRel, TRel], Q \\
\text{and } & A2: R \leq [\cdot R < SRel, TRel], Q \\
\text{and } & A3: (P, R) \notin (\text{indRelRSTPO} SRel \text{ TRel}) \land A4: (Q, R) \notin (\text{indRelRSTPO} SRel \text{ TRel})
\end{align*}

\begin{itemize}
\item \textbf{next}
\item \textbf{case (TargetTerm TP)}
\item \textbf{assume TP} ∈ T P
\item \textbf{with reflT} show (P, P) ∈ symcl (\text{indRelRSTPO} SRel \text{ TRel})
\item \textbf{unfolding} refl-on-def
\item \textbf{by} (simp add: symcl-def indRelRSTPO.target)
\end{itemize}

\begin{itemize}
\item \textbf{qed}
\item \textbf{qed}
\end{itemize}

\begin{itemize}
\item \textbf{moreover have} trans (\text{symcl} (\text{indRelRSTPO} SRel \text{ TRel}))
\item \textbf{proof} –
\item \textbf{have} \forall P Q R. P \leq [\cdot R < SRel, TRel], Q \land R \leq [\cdot R < SRel, TRel], Q \land (P, R) \notin (\text{indRelRSTPO} SRel \text{ TRel}) \\
\rightarrow Q \leq [\cdot R < SRel, TRel] \land P \lor Q \leq [\cdot R < SRel, TRel], R
\end{itemize}

\begin{itemize}
\item \textbf{proof} (cases P)
\item \textbf{case (SourceTerm SP)}
\item \textbf{assume A5: SP} ∈ S P
\item \textbf{show} Q \leq [\cdot R < SRel, TRel], P
\item \textbf{proof} (cases Q)
\item \textbf{case (SourceTerm SQ)}
\item \textbf{assume A6: SQ} ∈ S Q
\item \textbf{with} transS A1 A5 \textbf{ have} (SP, SQ) ∈ SRel
\item \textbf{using} \text{indRelRSTPO-to-SRel-and-TRel}(1)\textbf{[where] SRel=SRel and TRel=TRel} \\
\text{trancl-id[of SRel]}
\item \textbf{by} blast
\item \textbf{with symmS A5 A6} \textbf{ show} Q \leq [\cdot R < SRel, TRel], P
\item \textbf{unfolding} sym-def
\item \textbf{by} (simp add: indRelRSTPO.source)
\end{itemize}

\begin{itemize}
\item \textbf{next}
\item \textbf{case (TargetTerm TQ)}
\item \textbf{assume A6: TQ} ∈ T Q
\item \textbf{show} Q \leq [\cdot R < SRel, TRel], P
\item \textbf{proof} (cases R)
\item \textbf{case (SourceTerm SR)}
\item \textbf{assume A7: SR} ∈ S R
\item \textbf{with} fullAbs A2 A6 \textbf{ have} ([SR], TQ) ∈ TRel∗
\item \textbf{using} full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)\textbf{[where] SRel=SRel} \\
\text{and TRel=TRel} \text{ trancl-id[of TRel∗]} \text{ reflcl-of-refl-rel[of TRel]}
\item \textbf{unfolding} trancl-refcl[of TRel]
\item \textbf{by} blast
\item \textbf{with} transT reflT \textbf{ have} ([SR], TQ) ∈ TRel
\item \textbf{using} trancl-id[of TRel∗] \text{ reflcl-of-refl-rel[of TRel] trancl-refcl[of TRel]}
\item \textbf{by} auto
\item \textbf{with} symmT \textbf{ have} (TQ, [SR]) ∈ TRel
\item \textbf{unfolding} sym-def
\item \textbf{by} simp
\item \textbf{moreover from} fullAbs A1 A5 A6 \textbf{ have} ([SP], TQ) ∈ TRel∗
\item \textbf{using} full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2)\textbf{[where] SRel=SRel} \\
\text{and TRel=TRel} \\
\item \textbf{unfolding} trancl-refcl[of TRel]
\item \textbf{by} blast
\item \textbf{with} transT reflT \textbf{ have} ([SP], TQ) ∈ TRel
\item \textbf{using} trancl-id[of TRel∗] \text{ reflcl-of-refl-rel[of TRel] trancl-refcl[of TRel]}
\item \textbf{by} auto
\end{itemize}
ultimately have \([\{SP\}, \{SR\}] \in T\mathcal{R}el\)
using \textit{transT}
unfolding \textit{trans-def}
by \textit{blast}
with \textit{fullAbs} have \((SP, SR) \in S\mathcal{R}el\)
by \textit{simp}
with \textit{A3 A5 A7} show \textit{thesis}
by (\textit{simp add: indRelRSTPO.source})
\textbf{next}
case \(\textit{(TargetTerm TR)}\)
assume \(A7: TR \in T\mathcal{R}\)
with \textit{transT} \(A2 A6\) have \((TR, TQ) \in T\mathcal{R}el\)
using \textit{indRelRSTPO-to-SRel-and-TRel(4)}\[\textit{where SRel=SRel and TRel=TRel}\]
\textit{trancl-id} of \(T\mathcal{R}el\)
by \textit{blast}
with \textit{symmT} have \((TQ, TR) \in T\mathcal{R}el\)
unfolding \textit{sym-def}
by \textit{simp}
with \textit{A4 A6 A7} show \textit{thesis}
by (\textit{simp add: indRelRSTPO.target})
\textit{qed}
\textit{qed}
\textbf{next}
case \(\textit{(TargetTerm TP)}\)
assume \(A5: TP \in T\mathcal{P}\)
show \(Q \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P\)
\textbf{proof} (\textit{cases} \(Q\))
\textit{case} \(\textit{(SourceTerm SQ)}\)
assume \(SQ \in S\mathcal{Q}\)
with \textit{A1 A5} show \textit{thesis}
using \textit{indRelRSTPO-to-SRel-and-TRel(3)}\[\textit{where SRel=SRel and TRel=TRel}\]
by \textit{blast}
\textbf{next}
case \(\textit{(TargetTerm TQ)}\)
assume \(A6: TQ \in T\mathcal{Q}\)
with \textit{transT} \(A1 A5\) have \((TP, TQ) \in T\mathcal{R}el\)
using \textit{indRelRSTPO-to-SRel-and-TRel(4)}\[\textit{where SRel=SRel and TRel=TRel}\]
\textit{trancl-id} of \(T\mathcal{R}el\)
by \textit{blast}
with \textit{symmT} have \((TQ, TP) \in T\mathcal{R}el\)
unfolding \textit{sym-def}
by \textit{simp}
with \textit{A5 A6} show \(Q \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P\)
by (\textit{simp add: indRelRSTPO.target})
\textit{qed}
\textit{qed}
\textbf{moreover}
have \(\forall P Q R. P \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > Q \wedge P \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > R \wedge (Q, R) \notin (\textit{indRelRSTPO SRel TRel}) \rightarrow Q \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P \vee R \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P\)
\textbf{proof} \textit{clarify}
\textit{fix} \(P Q R\)
\textbf{assume} \(A1: P \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > Q \text{ and } A2: P \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > R \)
\textbf{and} \(A3: (Q, R) \notin (\textit{indRelRSTPO SRel TRel}) \text{ and } A4: (R, P) \notin (\textit{indRelRSTPO SRel TRel})\)
\textbf{show} \(Q \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P\)
\textbf{proof} (\textit{cases} \(P\))
\textit{case} \(\textit{(SourceTerm SP)}\)
\textbf{assume} \(A5: SP \in S\mathcal{P}\)
\textbf{show} \(Q \preceq [\cdot] R < S\mathcal{R}el, T\mathcal{R}el > P\)
\textbf{proof} (\textit{cases} \(Q\))
\textit{case} \(\textit{(SourceTerm SQ)}\)
assume $A6: SQ \in S Q$
with $\text{trans}S A1 A5$ have $(SP, SQ) \in SRel$
using $\text{indRelRSTPO-to-SRel-and-TRel(1) [where SRel=SRel and TRel=TRel]}$
$\text{trancl-id [of SRel]}$
by blast
with $\text{symmS} A5 A6$ show $Q \triangleq [\cdot] R < SRel, TRel > P$
unfolding $\text{sym-def}$
by $(\text{simp add: indRelRSTPO.source})$

next
case $(\text{TargetTerm} \ TQ)$
assume $A6: TQ \in T Q$
show $Q \triangleq [\cdot] R < SRel, TRel > P$
proof (cases $R$)
case $(\text{SourceTerm} \ SR)$
assume $A7: SR \in S R$
with $\text{trans}S A2 A5$ have $(SP, SR) \in SRel$
using $\text{indRelRSTPO-to-SRel-and-TRel(1) [where SRel=SRel and TRel=TRel]}$
$\text{trancl-id [of SRel]}$
by blast
with $\text{symmS} (SR, SP) \in SRel$
unfolding $\text{sym-def}$
by simp
with $A4 A5 A7$ show $\text{thesis}$
by $(\text{simp add: indRelRSTPO.source})$

next
case $(\text{TargetTerm} \ TR)$
from $\text{fullAbs} A1 A5 A6$ have $([SP], TQ) \in TRel^*$
using $\text{full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2) [where SRel=SRel and TRel=TRel]}$
$\text{unfolding} \ trans-def$
by blast
with $\text{transT} \ reflT$ have $([SP], TQ) \in TRel$
using $\text{transcl-id [of TRel^*]} \ \text{reflcl-of-refl-rel [of TRel]} \ \text{transcl-reflcl [of TRel]}$
by auto
with $\text{symmT} \ TQ \in TRel$
unfolding $\text{sym-def}$
by simp
moreover assume $A7: TR \in T R$
with $\text{fullAbs} A2 A5$ have $([SP], TR) \in TRel^*$
using $\text{full-abstraction-impl-indRelRSTPO-to-SRel-and-TRel(2) [where SRel=SRel and TRel=TRel]}$
$\text{unfolding} \ trans-def$
by blast
with $\text{transT} \ reflT$ have $([SP], TR) \in TRel$
using $\text{transcl-id [of TRel^*]} \ \text{reflcl-of-refl-rel [of TRel]} \ \text{transcl-reflcl [of TRel]}$
by auto
ultimately have $(TQ, TR) \in TRel$
using $\text{transT}$
unfolding $\text{trans-def}$
by blast
with $A3 A6 A7$ show $\text{thesis}$
by $(\text{simp add: indRelRSTPO.target})$
qued

next
case $(\text{TargetTerm} \ TP)$
assume $A5: TP \in T P$
show $Q \triangleq [\cdot] R < SRel, TRel > P$
proof (cases $Q$)
case $(\text{SourceTerm} \ SQ)$
assume $SQ \in S Q$
with A1 A5 show ?thesis
  using indRelRSTPO-to-SRel-and-TRel(3)[where SRel=SRel and TRel=TRel]
  by blast
next
case (TargetTerm TQ)
assume A6: TQ ∈ T Q
with transT A1 A5 have (TP, TQ) ∈ TRel
  using indRelRSTPO-to-SRel-and-TRel(4)[where SRel=SRel and TRel=TRel]
  trancl-id[of TRel]
  by blast
with symmT have (TQ, TP) ∈ TRel
  unfolding symm-def
  by simp
with A5 A6 show Q ≲ R[SRel,TRel] P
  by (simp add: indRelRSTPO_target)
qed
qed
moreover from reflS reflT have refl (indRelRSTPO SRel TRel)
  using indRelRSTPO-refl[where SRel=SRel and TRel=TRel]
  refl-rtrancl[of TRel]
  unfolding refl-on-def
  by auto
moreover have sym (indRelRSTPO SRel TRel)
  using sym-symcl[of symcl (TRel"=")]
  by simp
moreover have trans (indRelRSTPO SRel TRel)
  using symcl (indRelRSTPO SRel TRel)
  unfolding preorder-on-def
  by blast
ultimately show ?thesis
  using fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder
  [where SRel=(symcl (SRel"="))" and TRel=(symcl (TRel"=")) fullAbs
  refl-sym-closure-is-symm-closure
  unfolding preorder-on-def
  by blast
qed

lemma (in encoding) fully-abstract-impl-symcl-source-target-relation-is-preorder:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes fullAbs: fully-abstract ((symcl (SRel"="))" ((symcl (TRel"="))")
  shows ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ ((symcl (SRel"="))") = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel}
    ∧ ((symcl (TRel"="))") = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel}
    ∧ preorder (symcl Rel)
proof
  have refl ((symcl (TRel"="))")
    using refl-sym-cl-trans-closure-is-symm-refl-trans-closure[of TRel]
    refl-rtrncl[of TRel]
    unfolding symm-def refl-on-def
    by auto
  moreover have sym ((symcl (TRel"="))")
    using sym-symcl[of symcl (TRel"=")] sym-trncl[of symcl (TRel"=")]
    by simp
  moreover have trans ((symcl (TRel"="))")
    by simp
  ultimately show ?thesis
    using fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder[where
    SRel=(symcl (SRel"="))" and TRel=(symcl (TRel"=")) fullAbs
    refl-sym-closure-is-symm-refl-closure
    unfolding preorder-on-def
    by blast
qed
lemma (in encoding) fully-abstract-wrt-preorders-impl-source-target-relation-is-trans:
  fixes SRel :: ('procS × 'procS) set
  and TRel :: ('procT × 'procT) set
  assumes fullAbs: fully-abstract SRel TRel
shows ∃ Rel. (∀ S. (SourceTerm S, TargetTerm (|[S]|)) ∈ Rel)
  ∧ SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
  ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
  ∧ ((refl SRel ∧ trans TRel)
    ⟷ trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q}))
proof -
def rel: Rel ≡ (indRelSTEQ SRel TRel) − {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q}
  ∪ {(P, Q). ∃ S1 S2. S1 ∈ S P ∧ S2 ∈ S Q ∧ (S1, S2) ∉ SRel}
  ∪ {(P, Q). ∃ T1 T2. T1 ∈ T P ∧ T2 ∈ T Q ∧ (T1, T2) ∉ TRel})
from rel have ∀ S. (SourceTerm S, TargetTerm (|[S]|)) ∈ Rel
  by (simp add: indRelSTEQ.encL[where SRel=SRel and TRel=TRel])
moreover from rel have SRel = {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
proof auto
  fix S1 S2
  assume (S1, S2) ∈ SRel
  thus SourceTerm S1 ∈ SRel SourceTerm S2
    by (simp add: indRelSTEQ.source[where SRel=SRel and TRel=TRel])
qed
moreover from rel have TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
proof auto
  fix T1 T2
  assume (T1, T2) ∈ TRel
  thus TargetTerm T1 ∈ TRel TargetTerm T2
    by (simp add: indRelSTEQ.target[where SRel=SRel and TRel=TRel])
qed
moreover
  have (refl SRel ∧ trans TRel) ⟷ trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
proof (rule ifI, erule conjE)
  assume reflS: refl SRel and transT: trans TRel
  have Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q} = indRelSTEQ SRel TRel
  proof (auto simp add: rel)
    fix S
    show TargetTerm (|[S]|) ∈ SRel TargetTerm S
      by (rule indRelSTEQ.encL)
next
  fix S1 S2
  assume SourceTerm S1 ∈ SRel SourceTerm S2
  with fullAbs reflS transT have (S1, S2) ∈ SRel
    using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(1)[where SRel=SRel and TRel=TRel]
    by blast
moreover assume (S1, S2) ∉ SRel
  ultimately show False
    by simp
next
  fix T1 T2
  assume TargetTerm T1 ∈ TRel TargetTerm T2
  with fullAbs reflS transT have (T1, T2) ∈ TRel
    using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(5)[where SRel=SRel and TRel=TRel]
    by blast
moreover assume (T1, T2) ∉ TRel
  ultimately show False
    by simp
qed
thus trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
  using indRelSTEQ-trans[where SRel=SRel and TRel=TRel]
unfolding trans-def
by blast

next
assumes fullAbs
shows ?thesis
ultimately show qed

fixes SRel :: (Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \})

proof auto
fix S
from rel have (SourceTerm S, TargetTerm ([S])) ∈ Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \}
    by (simp add: indRelSTEPQ.encR)
moreover have (TargetTerm ([S]), SourceTerm S) ∈ Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \}
    by simp
ultimately have (SourceTerm S, SourceTerm S) ∈ Rel
    using transR
    unfolding trans-def
    by blast
with rel show (S, S) ∈ SRel
    by simp

next
fix TP TQ TR
assumes (TP, TQ) ∈ TRel
with rel have (TargetTerm TP, TargetTerm TQ) ∈ Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \}
    by (simp add: indRelSTEPQ.target)
moreover assume (TQ, TR) ∈ TRel
with rel have (TargetTerm TQ, TargetTerm TR) ∈ Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \}
    by (simp add: indRelSTEPQ.target)
ultimately have (TargetTerm TP, TargetTerm TR) ∈ Rel
    using transT
    unfolding trans-def
    by blast
with rel show (TP, TR) ∈ TRel
    by simp
qed

lemma (in encoding) fully-abstract-wrt-preorders-impl-source-target-relation-is-trans-B:
fixes SRel :: (’procS × ’procS) set
and TRel :: (’procT × ’procT) set
assumes fullAbs : fully-abstract SRel TRel
and reflT : refl TRel
and transT : trans TRel
shows \exists Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
    ∧ SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel\}
    ∧ TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel\}
    ∧ trans (Rel ∪ \{ (P, Q) . ∃ S. [S] ∈ T P ∧ S ∈ S Q \})
proof –
def rel : Rel⇒(indRelSTEPQ SRel TRel) = \{ (P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q \}
from fullAbs reflT have reflS : refl SRel
    unfolding refl-on-def
    by auto
from rel have \forall S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
    by (simp add: indRelSTEPQ.encR[where SRel=SRel and TRel=TRel])
moreover from rel have SRel = \{(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel\}
proof auto
fix S1 S2
assume (S1, S2) ∈ SRel
thus SourceTerm S1 ∼[\_]<SRel,TRel> SourceTerm S2
    by (simp add: indRelSTEPQ.source[where SRel=SRel and TRel=TRel])

225
next
fix S1 S2
assume SourceTerm S1 ~[\cdot]<SRel,TRel> SourceTerm S2
with fullAbs transT relS show (S1, S2) ∈ SRel
  using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(1)[where SRel=SRel
and TRel=TRel]
by blast
qed

moreover from rel have TRel = {((T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
proof auto
fix T1 T2
assume (T1, T2) ∈ TRel
thus TargetTerm T1 ~[\cdot]<SRel,TRel> TargetTerm T2
  by (simp add: indRelSTEQ.target[where SRel=SRel and TRel=TRel])
next
fix T1 T2
assume TargetTerm T1 ~[\cdot]<SRel,TRel> TargetTerm T2
with fullAbs transT relS show (T1, T2) ∈ TRel
  using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(5)[where SRel=SRel
and TRel=TRel]
by blast
qed

moreover from rel have Rel ∪ {((P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q} = indRelSTEQ SRel TRel
by (auto simp add: indRelSTEQ.encL)
hence trans (Rel ∪ {((P, Q). ∃ S. [S] ∈ T P ∧ S ∈ S Q})
  using indRelSTEQ.trans[where SRel=SRel and TRel=TRel]
unfolding trans-def
by auto
ultimately show \?thesis
by blast
qed

Thus an encoding is fully abstract w.r.t. an equivalence SRel on the source and an equivalence TRel on the target iff there exists a relation that relates source terms and their literal translations, whose sym
closure is a preorder such that the reduction of this sym closure to source/target terms is SRel/TRel.

lemma (in encoding) fully-abstract-wrt-equivalences-iff-symcl-source-target-relation-is-preorder:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
shows (fully-abstract SRel TRel ∧ equivalence TRel) =
(∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ SRel = {((S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel})
∧ TRel = {((T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel})
∧ preorder (symcl Rel))
proof (rule iffI)
assume fully-abstract SRel TRel ∧ equivalence TRel
thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ SRel = {((S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel})
∧ TRel = {((T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel})
∧ preorder (symcl Rel)
using fully-abstract-wrt-equivalences-impl-symcl-source-target-relation-is-preorder[where
SRel=SRel and TRel=TRel]
unfolding equiv-def
by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ SRel = {((S1, S2). (SourceTerm S1, SourceTerm S2) ∈ symcl Rel})
∧ TRel = {((T1, T2). (TargetTerm T1, TargetTerm T2) ∈ symcl Rel})
∧ preorder (symcl Rel)
from this obtain Rel
where ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and $\text{SRel} = \{(S_1, S_2), (\text{SourceTerm} S_1, \text{SourceTerm} S_2) \in \text{symcl Rel}\}$

and $\text{A1}: \text{TRel} = \{(T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{symcl Rel}\}$

and $\text{A2}: \text{preorder} (\text{symcl Rel})$

by blast

hence $\text{A5: fully-abstract SRel TRel}$

using source-target-relation-with-trans-symcl-impl-full-abstraction[where \text{Rel=}\text{Rel}]

unfolding preorder-on-def

by blast

moreover have equivalence TRel

unfolding trans-def equiv-def sym-def refl-on-def

proof auto

fix T

from $\text{A1 A2}$ show $(T, T) \in \text{TRel}$

unfolding preorder-on-def refl-on-def

by blast

next

fix $T_1 T_2$

assume $(T_1, T_2) \in \text{TRel}$

with $\text{A1}$ show $(T_2, T_1) \in \text{TRel}$

by (auto simp add: symcl-def)

next

fix $T_1 T_2 T_3$

assume $(T_1, T_2) \in \text{TRel}$ and $(T_2, T_3) \in \text{TRel}$

with $\text{A1 A2}$ show $(T_1, T_3) \in \text{TRel}$

unfolding trans-def preorder-on-def

by blast

qed

ultimately show fully-abstract SRel TRel $\land$ equivalence TRel

by blast

qed

lemma (in encoding) fully-abstract-iff-symcl-source-target-relation-is-preorder:

fixes SRel :: ('procS $\times$ 'procS) set

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

shows fully-abstract ((symcl (SRel$^-$))$^+$) ((symcl (TRel$^-$))$^+$) $=$

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

$\land (\text{symcl (SRel$^-$))$^+$} = \{(S_1, S_2), (\text{SourceTerm} S_1, \text{SourceTerm} S_2) \in \text{symcl Rel}\}$

$\land (\text{symcl (TRel$^-$))$^+$} = \{(T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{symcl Rel}\}$

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

proof (rule iff)

assume fully-abstract ((symcl (SRel$^-$))$^+$) ((symcl (TRel$^-$))$^+$)

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

$\land (\text{symcl (SRel$^-$))$^+$} = \{(S_1, S_2), (\text{SourceTerm} S_1, \text{SourceTerm} S_2) \in \text{symcl Rel}\}$

$\land (\text{symcl (TRel$^-$))$^+$} = \{(T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{symcl Rel}\}$

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

using fully-abstract-impl-symcl-source-target-relation-is-preorder[where SRel=SRel and TRel=TRel]

by blast

next

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

$\land (\text{symcl (SRel$^-$))$^+$} = \{(S_1, S_2), (\text{SourceTerm} S_1, \text{SourceTerm} S_2) \in \text{symcl Rel}\}$

$\land (\text{symcl (TRel$^-$))$^+$} = \{(T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{symcl Rel}\}$

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

from this obtain Rel

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

and (symcl (SRel$^-$))$^+$ = \{(S_1, S_2), (\text{SourceTerm} S_1, \text{SourceTerm} S_2) \in \text{symcl Rel}\}$

and $\text{A1}: (\text{symcl (TRel$^-$))$^+$} = \{(T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2) \in \text{symcl Rel}\}$

and $\text{A2}: \text{preorder} (\text{symcl Rel})$

by blast

thus fully-abstract ((symcl (SRel$^-$))$^+$) ((symcl (TRel$^-$))$^+$)

using source-target-relation-with-trans-symcl-impl-full-abstraction[where \text{Rel=}\text{Rel}]

227
9.5 Full Abstraction without Relating Translations to their Source Terms

Let \( \text{Rel} \) be the result of removing from \( \text{indRelSTEQ} \) all pairs of two source or two target terms that are not contained in \( \text{SRel} \) or \( \text{TRel} \). Then a fully abstract encoding ensures that \( \text{Rel} \) is trans iff \( \text{SRel} \) is refl and \( \text{TRel} \) is trans.

**Lemma (in encoding) full-abstraction-impl-indRelSTEQ-is-trans:**

- **Definition:** \( \text{SRel} :: (\text{'procS} \times \text{'procS}) \) set
- **Definition:** \( \text{TRel} :: (\text{'procT} \times \text{'procT}) \) set
- **Definition:** \( \text{Rel} :: ((\text{'procS}, \text{'procT}) \text{Proc} \times (\text{'procS}, \text{'procT}) \text{Proc}) \) set
- **Assumes:** \( \text{fullAbs: fully-abstract SRel TRel} \)
- **Assumes:** \( \text{rel} \)

**Proof**

By **auto**

**Proof auto**

- **Assume A1:** \( \text{refl SRel and A2: } \forall x \ y. \ (x, y) \in \text{TRel} \longmapsto (\forall z. \ (y, z) \in \text{TRel} \longmapsto (x, z) \in \text{TRel}) \)
- **Assume A3:** \( (P, Q) \in \text{Rel and A4: } (Q, R) \in \text{Rel} \)

**From fullAbs rel have A5:** \( \forall SP SQ. \ (\text{SourceTerm SP}, \text{SourceTerm SQ}) \in \text{Rel} \longmapsto ([SP], [SQ]) \in \text{TRel} \)

**By simp**

**From rel have A6:** \( \forall TP TQ. \ (\text{TargetTerm TP}, \text{TargetTerm TQ}) \in \text{Rel} \longmapsto (TP, TQ) \in \text{TRel} \)

**By simp**

**Have A7:** \( \forall SP TQ. \ (\text{SourceTerm SP}, \text{TargetTerm TQ}) \in \text{Rel} \longmapsto ([SP], TQ) \in \text{TRel} \)

**Proof clarify**

**Fix SP TQ**

**With rel have SourceTerm SP \text{~[\[]<SRel,TRel> TargetTerm TQ}**

**By simp**

**With A1 A2 fullAbs show ([SP], TQ) \in TRel**

**Using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(\beta)[where SRel=SRel and TRel=TRel]**

**Unfolding trans-def**

**By blast**

**Qed**

**Have A8:** \( \forall TP SQ. \ (\text{TargetTerm TP}, \text{SourceTerm SQ}) \in \text{Rel} \longmapsto (TP, [SQ]) \in \text{TRel} \)

**Proof clarify**

**Fix TP SQ**

**With rel have TargetTerm TP \text{~[\[<SRel,TRel> SourceTerm SQ}**

**By simp**

**With A1 A2 fullAbs show (TP, [SQ]) \in TRel**

**Using full-abstraction-wrt-preorders-impl-indRelSTEQ-to-SRel-and-TRel(\delta)[where SRel=SRel and TRel=TRel]**

**Unfolding trans-def**

**By blast**

**Qed**

**Show (P, R) \in Rel**

**Proof (cases P)**

**Case (SourceTerm SP)**

**Assume A9:** \( SP \in S P \)

**Show (P, R) \in Rel**

**Proof (cases Q)**

**Case (SourceTerm SQ)**

**Assume A10:** \( SQ \in S Q \)

228
with A3 A5 A9 have A11: ([SP], [SQ]) ∈ TRel
  by simp
show (P, R) ∈ Rel
proof (cases R)
case (SourceTerm SR)
  assume A12: SR ∈ S R
  with A4 A5 A10 have ([SQ], [SR]) ∈ TRel
    by simp
  with A2 A11 have ([SP], [SR]) ∈ TRel
    by blast
  with fullAbs have (SP, SR) ∈ SRel
    by simp
  with rel A9 A12 show (P, R) ∈ Rel
    by simp
next
case (TargetTerm TR)
  assume A12: TR ∈ T R
  from A9 have P ~[\[
    \]]<SRel,TRel> TargetTerm ([SP])
    by (simp add: indRelSTEQ.encR)
moreover from A4 A7 A10 A12 have ([SQ], TR) ∈ TRel
  by simp
  with A2 A11 have ([SP], TR) ∈ TRel
    by blast
  with A12 have TargetTerm ([SP]) ~[\[
    \]]<SRel,TRel> R
    by (simp add: indRelSTEQ_target)
ultimately have P ~[\[
    \]]<SRel, TRel> R
    by (rule indRelSTEQ.trans)
  with rel A9 A12 show (P, R) ∈ Rel
    by simp
qed
next
case (TargetTerm TQ)
  assume A10: TQ ∈ T Q
  with A3 A7 A9 have A11: ([SP], TQ) ∈ TRel
    by simp
show (P, R) ∈ Rel
proof (cases R)
case (SourceTerm SR)
  assume A12: SR ∈ S R
  with A4 A6 A10 have (TQ, [SR]) ∈ TRel
    by simp
  with A2 A11 have ([SP], [SR]) ∈ TRel
    by blast
  with fullAbs have (SP, SR) ∈ SRel
    by simp
  with rel A9 A12 show (P, R) ∈ Rel
    by simp
next
case (TargetTerm TR)
  assume A12: TR ∈ T R
  from A9 have P ~[\[
    \]]<SRel,TRel> TargetTerm ([SP])
    by (simp add: indRelSTEQ.encR)
moreover from A4 A6 A10 A12 have (TQ, TR) ∈ TRel
  by simp
  with A2 A11 have ([SP], TR) ∈ TRel
    by blast
  with A12 have TargetTerm ([SP]) ~[\[
    \]]<SRel,TRel> R
    by (simp add: indRelSTEQ_target)
ultimately have P ~[\[
    \]]<SRel, TRel> R
    by (rule indRelSTEQ.trans)
  with A9 A12 rel show (P, R) ∈ Rel

by simp
qed
qed
next


case (TargetTerm TP)
assume A9: TP ∈ T P
show (P, R) ∈ Rel
proof (cases Q)
case (SourceTerm SQ)
assume A10: SQ ∈ S Q
with A3 A8 A9 have A11: (TP, [SQ]) ∈ TRel
  by simp
show (P, R) ∈ Rel
proof (cases R)
case (SourceTerm SR)
assume A12: SR ∈ S R
with A4 A5 A10 have ([SQ], [SR]) ∈ TRel
  by blast
with A9 have P ~[.]<SRel,TRel> TargetTerm ([SR])
  by (simp add: indRelSTEQ.target)
moreover from A12 have TargetTerm ([SR]) ~[.]<SRel,TRel> R
  by (simp add: indRelSTEQ.encL)
ultimately have P ~[.]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
with rel A9 A12 show (P, R) ∈ Rel
  by simp
next

case (TargetTerm TR)
assume A12: TR ∈ T R
with A4 A7 A10 have ([SQ], TR) ∈ TRel
  by simp
with A2 A11 have (TP, TR) ∈ TRel
  by blast
with rel A9 A12 show (P, R) ∈ Rel
  by simp
qed
next


case (TargetTerm TQ)
assume A10: TQ ∈ T Q
with A3 A6 A9 have A11: (TP, TQ) ∈ TRel
  by simp
show (P, R) ∈ Rel
proof (cases R)
case (SourceTerm SR)
assume A12: SR ∈ S R
with A4 A8 A10 have (TQ, [SR]) ∈ TRel
  by simp
with A2 A11 have (TP, [SR]) ∈ TRel
  by blast
with A9 have P ~[.]<SRel,TRel> TargetTerm ([SR])
  by (simp add: indRelSTEQ.target)
moreover from A12 have TargetTerm ([SR]) ~[.]<SRel,TRel> R
  by (simp add: indRelSTEQ.encL)
ultimately have P ~[.]<SRel,TRel> R
  by (rule indRelSTEQ.trans)
with rel A9 A12 show (P, R) ∈ Rel
  by simp
next

case (TargetTerm TR)
assume $A_{12} \colon TR \in TR$
with $A_4 A_6 A_{10}$ have $(TQ, TR) \in TRel$
by simp
with $A_2 A_{11}$ have $(TP, TR) \in TRel$
by blast
with $A_9 A_{12}$ rel show $(P, R) \in Rel$
by simp
qed

next
assume $B \colon \forall x y. (x, y) \in Rel \rightarrow (\forall z. (y, z) \in Rel \rightarrow (x, z) \in Rel)$
thus refl SRel
unfolding refl-on-def
proof auto
fix $S$
from rel have $(\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel$
by (simp add: indRelSTEQ.encR)
moreover from rel have $(\text{TargetTerm } ([S]), \text{SourceTerm } S) \in Rel$
by (simp add: indRelSTEQ.encL)
ultimately have $(\text{SourceTerm } S, \text{SourceTerm } S) \in Rel$
by blast
with rel show $(S, S) \in SRel$
by simp
qed

next
fix $TP$ $TQ$ $TR$
assume $\forall x y. (x, y) \in Rel \rightarrow (\forall z. (y, z) \in Rel \rightarrow (x, z) \in Rel)$
moreover assume $(TP, TQ) \in TRel$
with rel have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in Rel$
by simp
moreover assume $(TQ, TR) \in TRel$
with rel have $(\text{TargetTerm } TQ, \text{TargetTerm } TR) \in Rel$
by simp
ultimately have $(\text{TargetTerm } TP, \text{TargetTerm } TR) \in Rel$
by blast
with rel show $(TP, TR) \in TRel$
by simp
qed

Whenever an encoding induces a trans relation that includes $SRel$ and $TRel$ and relates source terms to their literal translations in both directions, the encoding is fully abstract w.r.t. $SRel$ and $TRel$.

lemma (in encoding) trans-source-target-relation-impl-fully-abstract:
fixes $Rel \ :: \ ((\text{procS} \times \text{procT}) \text{Proc} \times ((\text{procS} \times \text{procT}) \text{Proc}) \text{set}$
and $SRel \ :: \ (\text{procS} \times \text{procS}) \text{set}$
and $TRel \ :: \ (\text{procT} \times \text{procT}) \text{set}$
assumes $enc \ :: \ \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in Rel$
and $srel \ :: \ SRel = \{(S1, S2). (\text{SourceTerm } S1, \text{SourceTerm } S2) \in Rel\}$
and $trel \ :: \ TRel = \{(T1, T2). (\text{TargetTerm } T1, \text{TargetTerm } T2) \in Rel\}$
and $\text{trans} \ :: \ \text{trans} \ \text{Rel}$
shows fully-abstract $SRel$ $TRel$
proof auto
fix $S1$ $S2$
assume $(S1, S2) \in SRel$
with $srel$ have $(\text{SourceTerm } S1, \text{SourceTerm } S2) \in Rel$
by simp
with $enc$ $\text{trans}$ have $(\text{TargetTerm } ([S1]), \text{TargetTerm } ([S2])) \in Rel$
unfolding $\text{trans-def}$
by blast
with trel show ([S1], [S2]) ∈ TRel
by simp
next
fix S1 S2
assume ([S1], [S2]) ∈ TRel
with trel have (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
by simp
with enc trans have (SourceTerm S1, SourceTerm S2) ∈ Rel
unfolding trans-def
by blast
with srel show (S1, S2) ∈ SRel
by simp
qed

Assume TRel is a preorder. Then an encoding is fully abstract w.r.t. SRel and TRel if 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 × ('procS, 'procT) Proc) set
assumes enc: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and trans: trans (Rel ∪ {(P, Q). ∃ S. [S] ∈ T P & S ∈ S Q})
shows fully-abstract {(S1, S2). (SourceTerm S1, SourceTerm S2) ∈ Rel}
{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
proof auto
fix S1 S2
def rel': Rel'≡Rel ∪ {(P, Q). ∃ S. [S] ∈ T P & S ∈ S Q}
from rel' have (TargetTerm ([S1]), SourceTerm S1) ∈ Rel'
by simp
moreover assume (SourceTerm S1, SourceTerm S2) ∈ Rel'
with rel' have (SourceTerm S1, SourceTerm S2) ∈ Rel'
by simp
moreover from enc rel' have (SourceTerm S2, TargetTerm ([S2])) ∈ Rel'
by simp
ultimately show (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel
using trans rel'
unfolding trans-def
by blast
next
fix S1 S2
def rel': Rel'≡Rel ∪ {(P, Q). ∃ S. [S] ∈ T P & S ∈ S Q}
from enc rel' have (SourceTerm S1, TargetTerm ([S1])) ∈ Rel'
by simp
moreover assume (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel'
with rel' have (TargetTerm ([S1]), TargetTerm ([S2])) ∈ Rel'
by simp
moreover from rel' have (TargetTerm ([S2]), SourceTerm S2) ∈ Rel'
by simp
ultimately show (SourceTerm S1, SourceTerm S2) ∈ Rel
using trans rel'
unfolding trans-def
by blast
qed

lemma (in encoding) fully-abstract-wrt-preorders-iff-source-target-relation-is-transB:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes preord: preorder TRel
shows fully-abstract SRel TRel =
\( (\exists \ SourceTerm S, TargetTerm ([S])) \in Rel \)
\[ \land \ SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel \]
\[ \land \ TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel \]
\[ \land \ trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q\}) \]

**proof (rule ifI)**

**assume** fully-abstract SRel TRel

**with** proof **show** \( \exists \ SourceTerm S, TargetTerm ([S])) \in Rel \)
\[ \land \ SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel \]
\[ \land \ TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel \]
\[ \land \ trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q\}) \]

**using** fully-abstract-upto-preorders-impl-source-target-relation-is-trans[where SRel=SRel
and TRel=TRel]

**unfolding** preorder-on-def refl-on-def

**by** auto

**next**

**assume** \( \exists \ SourceTerm S, TargetTerm ([S])) \in Rel \)
\[ \land \ SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel \]
\[ \land \ TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel \]
\[ \land \ trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q\}) \]

**from this obtain** Rel

**where** \( \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \)
**and** SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel

**and** TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel

**and** trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q\})

**by** blast

**thus** fully-abstract SRel TRel

**using** source-target-relation-with-trans-impl-full-abstraction[where Rel=Rel]

**by** blast

**qed**

The same holds if to obtain transitivity the union may contain additional pairs that do neither relate two source nor two target terms.

**lemma (in encoding)** fully-abstract-upto-preorders-iff-source-target-relation-union-is-trans:

**fixes** SRel :: ('procS x 'procS) set

**and** TRel :: ('procT x 'procT) set

**shows** (fully-abstract SRel TRel \( \land \) refl SRel \( \land \) trans TRel) =

\( (\exists Rel. \ (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \)
\[ \land \ SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel \]
\[ \land \ TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel \]
\[ \land \ (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \iff Q \in ProcT)
\land \ trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q \cup Rel')\)) \]

**proof (rule ifI, (erule conjE)+)**

**assume** fully-abstract SRel TRel and refl SRel and trans TRel

**from this obtain** Rel **where** A1: \( \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \)

**and** A2: SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel

**and** A3: TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel

**and** A4: trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q \cup Rel')\)

**using** fully-abstract-upto-preorders-impl-source-target-relation-is-trans[where SRel=SRel
and TRel=TRel]

**by** blast

**have** \( \forall (P, Q) \in \{\}. P \in ProcS \iff Q \in ProcT \)

**by** simp

**moreover from** A4 **have** trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q \cup \{\})

**unfolding** trans-def

**by** blast

**ultimately show** \( \exists Rel. (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel \)
\[ \land \ SRel = \{(S_1, S_2), (SourceTerm S_1, SourceTerm S_2)\} \in Rel \]
\[ \land \ TRel = \{(T_1, T_2), (TargetTerm T_1, TargetTerm T_2)\} \in Rel \]
\[ \land \ (\exists Rel'. (\forall (P, Q) \in Rel'. P \in ProcS \iff Q \in ProcT)
\land \ trans(Rel \cup \{(P, Q), \exists S. [S] \in T P \land S \in S Q \cup Rel')\)) \]

233
using A1 A2 A3
by blast
next
assume \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge \text{SRel} = \{(S_1, S_2), (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}\} \)
\( \wedge \text{TRel} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\} \)
\( \wedge (\exists \text{Rel}'. (\forall (P, Q) \in \text{Rel}'. P \in \text{ProcS} \leftrightarrow Q \in \text{ProcT}) \)
\( \wedge \text{trans} (\text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}') \)
from this obtain \text{Rel} \text{Rel}'
where B1: \( \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \)
and B2: \( \text{SRel} = \{(S_1, S_2), (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel}\} \)
and B3: \( \text{TRel} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\} \)
and B4: \( \forall (P, Q) \in \text{Rel}'. P \in \text{ProcS} \leftrightarrow Q \in \text{ProcT} \)
and B5: \( \text{trans} (\text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}') \)
by blast
have fully-abstract \text{SRel} \text{TRel}
proof auto
fix \( S_1 S_2 \)
have \( (\text{TargetTerm } ([S_1]), \text{SourceTerm } S_1) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
moreover assume \( (S_1, S_2) \in \text{SRel} \)
with B2 have \( (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
moreover from B1
have \( (\text{SourceTerm } S_2, \text{TargetTerm } ([S_2])) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
ultimately have \( (\text{TargetTerm } ([S_1]), \text{TargetTerm } ([S_2])) \in \text{Rel} \cup \text{Rel}' \)
using B5
unfolding \text{trans-def}
by blast
with B3 B4 show \( ([S_1], [S_2]) \in \text{TRel} \)
by blast
next
fix \( S_1 S_2 \)
from B1
have \( (\text{SourceTerm } S_1, \text{TargetTerm } ([S_1])) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
moreover assume \( ([S_1], [S_2]) \in \text{TRel} \)
with B3
have \( (\text{TargetTerm } ([S_1]), \text{TargetTerm } ([S_2])) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
moreover
have \( (\text{TargetTerm } ([S_2]), \text{SourceTerm } S_2) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
ultimately have \( (\text{SourceTerm } S_1, \text{SourceTerm } S_2) \in \text{Rel} \cup \text{Rel}' \)
using B5
unfolding \text{trans-def}
by blast
with B2 B4 show \( (S_1, S_2) \in \text{SRel} \)
by blast
qed
moreover have \text{refl} \text{SRel}
unfolding \text{refl-on-def}
proof auto
fix \( S \)
from B1 have \( (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
moreover
have \( (\text{TargetTerm } ([S]), \text{SourceTerm } S) \in \text{Rel} \cup \{(P, Q), \exists S. [S] \in T \wedge S \in S \cup \text{Rel}' \)
by simp
ultimately have \( (\text{SourceTerm } S, \text{SourceTerm } S) \in \text{Rel} \cup \text{Rel}' \)
using \( B5 \)
unfolding trans-def
by blast
with \( B2 \ B4 \) show \((S, S) \in SRel\)
by blast
qed
moreover have trans TRel
unfolding trans-def
proof clarify
fix TP TQ TR
assume \((TP, TQ) \in TRel \text{ and } (TQ, TR) \in TRel\)
with \( B3 \ B4 \ B5 \) show \((TP, TR) \in TRel\)
unfolding trans-def
by blast
qed
ultimately show fully-abstract SRel TRel \(\land\) refl SRel \(\land\) trans TRel
by blast
qed

end

theory CombinedCriteria
  imports DivergenceReflection SuccessSensitiveness FullAbstraction OperationalCorrespondence
begin

10 Combining Criteria

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

lemma (in encoding) criterion-iff-source-target-relation-impl-indRelR:
  fixes Cond :: ('procS ⇒ 'procT) ⇒ bool
  assumes Cond enc = (∃Rel. ∀S. (SourceTerm S, TargetTerm (⟦S⟧)) ∈ Rel) \(\land\) Pred Rel
  shows Cond enc = (∃Rel'. Pred (indRelR \(\cup\) Rel'))
proof (rule iffI)
  assume Cond enc
  with assms obtain Rel where A1: \(\forall S. (SourceTerm S, TargetTerm (⟦S⟧)) \in Rel\) \(\land\) A2: Pred Rel
  by blast
  from A1 have Rel = indRelR \(\cup\) (Rel \(\land\) indRelR)
  by (auto simp add: indRelR.simps)
  with A2 have Pred (indRelR \(\cup\) (Rel \(\land\) indRelR))
  by simp
  thus ∃Rel'. Pred (indRelR \(\cup\) Rel')
  by blast
next
  assume ∃Rel'. Pred (indRelR \(\cup\) Rel')
  from this obtain Rel' where Pred (indRelR \(\cup\) Rel')
  by blast
  moreover have \(\forall S. (SourceTerm S, TargetTerm (⟦S⟧)) \in (indRelR \(\cup\) Rel')\)
  by (simp add: indRelR.encR)
  ultimately show Cond enc
  using assms
  by blast
qed

lemma (in encoding) combine-conditions-on-pairs-of-relations:
proof
lemma witness indRelR.
  
  literal translations. The following lemmata help us to combine such conditions by switching to the
  
  We mapped several criteria on conditions on relations that relate at least all source terms and their
  
  (lemma
    fixes RelA RelB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  
  assumes ∀ (P, Q) ∈ RelA. CondA (P, Q)
  and ∀ (P, Q) ∈ RelB. CondB (P, Q)
  shows (∀ (P, Q) ∈ RelA ∩ RelB. CondA (P, Q)) ∧ (∀ (P, Q) ∈ RelA ∩ RelB. CondB (P, Q))

  using assms
  by blast

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

  using assms
  by blast

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

  using assms
  by blast

We mapped several criteria on conditions on relations that relate at least all source terms and their
  
  literal translations. The following lemmata help us to combine such conditions by switching to the
  
  witness indRelR.

lemma (in encoding) combine-conditions-on-relations-indRelR:
  
  fixes Rel RelA RelB :: (('procS, 'procT) Proc × ('procS, 'procT) Proc) set
  
  assumes A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ RelA
  and A2: ∀ (P, Q) ∈ RelA. CondA (P, Q)
  and A3: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ RelB
  and A4: ∀ (P, Q) ∈ RelB. CondB (P, Q)
  shows ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q))
  ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q))
  ∧ (Cond indRelR ⇒ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q)) ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q)) ∧ Cond Rel)

  proof
    
    have A5: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ indRelR
    by (simp add: indRelR.encR)
    moreover have A6: indRelR ⊆ RelA
    proof clarify
      
      fix P Q
      assume (P, Q) ∈ indRelR
      from this A1 show (P, Q) ∈ RelA
      by (induct, simp)
      qed
      moreover have A7: indRelR ⊆ RelB
      proof clarify
        
        fix P Q
        assume (P, Q) ∈ indRelR
        from this A3 show (P, Q) ∈ RelB
        by (induct, simp)
        qed
ultimately show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q)) ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q))

using combine-conditions-on-sets-and-pairs-of-relations[where RelA=RelA and RelB=RelB and CondA=CondA and CondB=CondB and Rel=indRelR and Cond=λR. ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ R] A2 A4

by blast

from A2 A4 A5 A6 A7

show Cond indRelR ⇒ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ (∀ (P, Q) ∈ Rel. CondA (P, Q)) ∧ (∀ (P, Q) ∈ Rel. CondB (P, Q)) ∧ Cond Rel)

using combine-conditions-on-sets-and-pairs-of-relations[where RelA=RelA and RelB=RelB and CondA=CondA and CondB=CondB and Rel=indRelR and Cond=λR. ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ R ∧ Cond R]

by blast

qed

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

fixes PredA PredB :: (’procS, ’procT) Proc ⇒ bool

shows ((∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred PredA) ∧ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-reflects-pred PredB))

= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred PredA ∧ rel-reflects-pred PredB)

proof (rule iffI, erule conjE)

assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred PredA

from this obtain RelA where A1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ RelA

and A2: rel-reflects-pred RelA PredA

by blast

assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-reflects-pred PredB

from this obtain RelB where A3: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ RelB

and A4: rel-reflects-pred RelB PredB

by blast

from A2 have ∀ (P, Q) ∈ RelA. PredA P ↔ PredA Q

by blast

moreover from A4 have ∀ (P, Q) ∈ RelB. PredB Q → PredB P

by blast

ultimately have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ (∀ (P, Q) ∈ Rel. PredA P = PredA Q) ∧ (∀ (P, Q) ∈ Rel. PredB Q → PredB P)

using combine-conditions-on-relations-indRelR(1)[where RelA=RelA and RelB=RelB and CondA=λ(P, Q). PredA P ↔ PredA Q and CondB=λ(P, Q). PredB Q → PredB P] A1 A3

by simp

thus ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel PredA ∧ rel-reflects-pred Rel PredB

by blast

next

assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel PredA ∧ rel-reflects-pred Rel PredB

thus (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-respects-pred Rel PredA) ∧ (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel) ∧ rel-reflects-pred Rel PredB)

by blast

qed

10.1 Divergence Reflection and Success Sensitiveness

We combine results on divergence reflection and success sensitiveness to analyse their combined effect on an encoding function. An encoding is success sensitive and reflects divergence iff there exists a relation that relates source terms and their literal translations that reflects divergence and respects success.

lemma (in encoding-wrt-barbs) WSS-DR-iff-source-target-rel:

fixes success :: ’barbs

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

\text{proof} –

\text{have} \, \forall \text{Rel.} \, \text{rel-reflects-divergence Rel} (\text{STCal Source Target}) \\
= \text{rel-reflects-pred Rel divergentST}

\text{by} \, (\text{simp add: divergentST-} \text{STCal-divergent})

\text{moreover have} \, \forall \text{Rel.} \, (\text{rel-weakly-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \\
= \text{rel-respects-pred Rel} (\lambda P. \downarrow P, \downarrow \text{success})

\text{by} \, (\text{simp add: STCalWB-reachesBarbST})

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

\text{using} \, \text{success-sensitive-iff-source-target-rel-weakly-respects-success}(1) \\
\text{divergence-reflection-iff-source-target-rel-reflects-divergence}

\text{indRelR-cond-respects-predA-and-reflects-predB}[\text{where}]

\text{PredA} = \lambda P. \downarrow P, \downarrow \text{success} \quad \text{and} \quad \text{PredB} = \text{divergentST}

\text{by simp}

qed

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

\text{fixes success : 'barbs}

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

\text{proof} –

\text{have} \, \forall \text{Rel.} \, \text{rel-reflects-divergence Rel} (\text{STCal Source Target}) \\
= \text{rel-reflects-pred Rel divergentST}

\text{by} \, (\text{simp add: divergentST-} \text{STCal-divergent})

\text{moreover have} \, \forall \text{Rel.} \, (\text{rel-respects-barb-set Rel} (\text{STCalWB SWB TWB}) \{\text{success}\} \\
= \text{rel-respects-pred Rel} (\lambda P. \downarrow P, \downarrow \text{success})

\text{by} \, (\text{simp add: STCalWB-hasBarbST})

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

\text{using} \, \text{success-sensitive-iff-source-target-rel-weakly-respects-success}(1) \\
\text{divergence-reflection-iff-source-target-rel-reflects-divergence}

\text{indRelR-cond-respects-predA-and-reflects-predB}[\text{where}]

\text{PredA} = \lambda P. \downarrow P, \downarrow \text{success} \quad \text{and} \quad \text{PredB} = \text{divergentST}

\text{by simp}

qd

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 sensiti- 

\text{viveness, 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 sensitiv- 

eness.}

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

\text{fixes success : 'barbs}

\text{and TRel : ('procT × 'procT) set}

\text{assumes encRS: enc-weakly-respects-barb-set \{success\}}

\text{and trelPS: rel-weakly-preserves-barb-set TRel TWB \{success\}}

\text{and trelRS: rel-weakly-reflects-barb-set TRel TWB \{success\}}

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

\text{proof auto}

fix P Q P'
assume $P \leq [\cdot]RT <TRel> Q$ and $P \rightarrow (Calculus (STCalWB SWB TWB)) \ast P'$
and $P' \downarrow <STCalWB SWB TWB> success$
thus $Q \downarrow <STCalWB SWB TWB> success$

proof (induct arbitrary: $P'$)

next

\begin{enumerate}
\item case (encR $S$)
\begin{enumerate}
\item assume SourceTerm $S$ \rightarrow (Calculus (STCalWB SWB TWB)) \ast P' and $P' \downarrow <STCalWB SWB TWB> success$
\item hence $S \downarrow <SWB> success$
\item using STCalWB-reachesBarbST
\item by blast
\end{enumerate}
\end{enumerate}

\begin{enumerate}
\item with $encRS$ have $[S] \downarrow <TWB> success$
\item by simp
\item thus TargetTerm $([S]) \downarrow <STCalWB SWB TWB> success$
\item using STCalWB-reachesBarbST
\item by blast
\end{enumerate}

next

\begin{enumerate}
\item case (source $S$)
\begin{enumerate}
\item assume SourceTerm $S$ \rightarrow (Calculus (STCalWB SWB TWB)) \ast P' and $P' \downarrow <STCalWB SWB TWB> success$
\item thus SourceTerm $S \downarrow <STCalWB SWB TWB> success$
\item by blast
\end{enumerate}
\end{enumerate}

next

\begin{enumerate}
\item case (target $T1$ $T2$)
\begin{enumerate}
\item assume $(T1, T2) \in TRel$
\item moreover assume TargetTerm $T1$ \rightarrow (Calculus (STCalWB SWB TWB)) \ast P'
\item and $P' \downarrow <STCalWB SWB TWB> success$
\item hence $T1 \downarrow <TWB> success$
\item using STCalWB-reachesBarbST
\item by blast
\item ultimately have $T2 \downarrow <TWB> success$
\item using trelPS
\item by simp
\item thus TargetTerm $T2 \downarrow <STCalWB SWB TWB> success$
\item using STCalWB-reachesBarbST
\item by blast
\end{enumerate}
\end{enumerate}

next

\begin{enumerate}
\item case (trans $P$ $Q$ $R$)
\begin{enumerate}
\item assume $P \rightarrow (Calculus (STCalWB SWB TWB)) \ast P' and P' \downarrow <STCalWB SWB TWB> success$
\item and $\forall P', P \rightarrow (Calculus (STCalWB SWB TWB)) \ast P' \rightarrow P' \downarrow <STCalWB SWB TWB> success$
\item $\Rightarrow Q \downarrow <STCalWB SWB TWB> success$
\item hence $Q \downarrow <STCalWB SWB TWB> success$
\item by simp
\item moreover assume $\forall Q', Q \rightarrow (Calculus (STCalWB SWB TWB)) \ast Q' \rightarrow Q' \downarrow <STCalWB SWB TWB> success$
\item $\Rightarrow R \downarrow <STCalWB SWB TWB> success$
\item ultimately show $R \downarrow <STCalWB SWB TWB> success$
\item by blast
\end{enumerate}
\end{enumerate}

qed

next

\begin{enumerate}
\item fix $P$ $Q$ $Q'$
\item assume $P \leq [\cdot]RT <TRel> Q$ and $Q \rightarrow (Calculus (STCalWB SWB TWB)) \ast Q'$
\item and $Q' \downarrow <STCalWB SWB TWB> success$
\item thus $P \downarrow <STCalWB SWB TWB> success$
\item proof (induct arbitrary: $Q'$)
\item case (encR $S$)
\item assume TargetTerm $([S]) \rightarrow (Calculus (STCalWB SWB TWB)) \ast Q'$
\item and $Q' \downarrow <STCalWB SWB TWB> success$
\item hence $[S] \downarrow <TWB> success$
\item using STCalWB-reachesBarbST
\item by blast
\item with $encRS$ have $S \downarrow <SWB> success$
\item by simp
\item thus SourceTerm $S \downarrow <STCalWB SWB TWB> success$
\item using STCalWB-reachesBarbST
\end{enumerate}

239
by blast

next

  case (source S)
  assume SourceTerm S \mapsto (Calculus (STCalWB SWB TWB)):: Q' and Q' \downarrow<STCalWB SWB TWB> success
  thus SourceTerm S \downarrow<STCalWB SWB TWB> success
  by blast

next

  case (target T1, T2)
  assume (T1, T2) \in TRel
  moreover assume TargetTerm T2 \mapsto (Calculus (STCalWB SWB TWB)):: Q' and Q' \downarrow<STCalWB SWB TWB> success
  hence T2 \downarrow<STCalWB SWB TWB> success
  using STCalWB-reachesBarbST
  by blast
  ultimately have T1 \downarrow<STCalWB SWB TWB> success
  using trelRS
  by blast
  thus TargetTerm T1 \downarrow<STCalWB SWB TWB> success
  using STCalWB-reachesBarbST
  by blast

next

  case (trans P Q R R')
  assume R \mapsto (Calculus (STCalWB SWB TWB)):: R' and R' \downarrow<STCalWB SWB TWB> success
  and \bigwedge Q', R \mapsto (Calculus (STCalWB SWB TWB)):: R' \Longrightarrow R' \downarrow<STCalWB SWB TWB> success
  hence Q' \downarrow<STCalWB SWB TWB> success
  by simp
  moreover assume \bigwedge Q'. Q \mapsto (Calculus (STCalWB SWB TWB)):: Q' \Longrightarrow Q' \downarrow<STCalWB SWB TWB> success
  \quad \Longrightarrow P' \downarrow<STCalWB SWB TWB> success
  ultimately show P' \downarrow<STCalWB SWB TWB> success
  by blast

qed

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 auto

  fix P Q x P'
  assume P \leq\leq \exists RT <TRel> Q and P \mapsto (Calculus (STCalWB SWB TWB)):: P'
  and P' \downarrow<STCalWB SWB TWB> x
  thus Q' \downarrow<STCalWB SWB TWB> x

proof (induct arbitrary: P')

  case (encR S)
  assume SourceTerm S \mapsto (Calculus (STCalWB SWB TWB)):: P' and P' \downarrow<STCalWB SWB TWB> x
  hence S \downarrow<SWB> x
  using STCalWB-reachesBarbST
  by blast
  with encRS have \([S]\downarrow<TWB> x
  by simp
  thus TargetTerm ([S]) \downarrow<STCalWB SWB TWB> x
  using STCalWB-reachesBarbST
  by blast

next

  case (source S)
  assume SourceTerm S \mapsto (Calculus (STCalWB SWB TWB)):: P' and P' \downarrow<STCalWB SWB TWB> x
  thus SourceTerm S \downarrow<STCalWB SWB TWB> x
  by blast
next
case (target T1 T2)
assume (T1, T2) ∈ TRel
moreover assume TargetTerm T1 ↦−→ (Calculus (STCalWB SWB TWB))∗ P′
and P′↓<STCalWB SWB TWB>x
hence T1⇓<TWB>x
  using STCalWB-reachesBarbST
by blast
ultimately have T2⇓<TWB>x
  using STCalWB-reachesBarbST
by blast
thus TargetTerm T2⇓<STCalWB SWB TWB>x
next
case (trans P Q R)
assume P ↦−→ (Calculus (STCalWB SWB TWB))∗ P′ and P′↓<STCalWB SWB TWB>x
and \( \land P′. P ↦−→ (Calculus (STCalWB SWB TWB))∗ P′ \Rightarrow P′↓<STCalWB SWB TWB>x \)
⇒ Q↓<STCalWB SWB TWB>x
hence Q⇓<STCalWB SWB TWB>x
by simp
moreover assume \( \land Q′. Q ↦−→ (Calculus (STCalWB SWB TWB))∗ Q′ \Rightarrow Q′↓<STCalWB SWB TWB>x \)
⇒ R↓<STCalWB SWB TWB>x
ultimately show R⇓<STCalWB SWB TWB>x
by blast
qed
next
fix P Q x Q′
assume P ≲ [ ]<R T Rel> Q and Q ↦−→ (Calculus (STCalWB SWB TWB))∗ Q′
and Q′↓<STCalWB SWB TWB>x
thus P↓<STCalWB SWB TWB>x
proof (induct arbitrary: Q′)
case (encR S)
assume TargetTerm ([S]) ↦−→ (Calculus (STCalWB SWB TWB))∗ Q′
and [S]↓<TWB>x
hence [S]⇓<TWB>x
  using STCalWB-reachesBarbST
by blast
with encRS have S⇓<SWB>x
by simp
thus SourceTerm S⇓<STCalWB SWB TWB>x
  using STCalWB-reachesBarbST
by blast
next
case (source S)
assume SourceTerm S ↦−→ (Calculus (STCalWB SWB TWB))∗ Q′ and Q′↓<STCalWB SWB TWB>x
thus SourceTerm S⇓<STCalWB SWB TWB>x
by blast
next
case (target T1 T2)
assume (T1, T2) ∈ TRel
moreover assume TargetTerm T2 ↦−→ (Calculus (STCalWB SWB TWB))∗ Q′
and Q′↓<STCalWB SWB TWB>x
hence T2⇓<TWB>x
  using STCalWB-reachesBarbST
by blast
ultimately have T1⇓<TWB>x
  using STCalWB-reachesBarbST
by blast
thus TargetTerm T1⇓<STCalWB SWB TWB>x
  using STCalWB-reachesBarbST
by blast

next

case (trans $P \ Q \ R'$)
assume $R \rightarrow (Calculus (STCalWB SWB TWB)) \ast R' \text{ and } R' \downarrow<STCalWB SWB TWB>$
and $\bigwedge R'. R \rightarrow (Calculus (STCalWB SWB TWB)) \ast R' \Rightarrow R' \downarrow<STCalWB SWB TWB>$

hence $Q \downarrow<STCalWB SWB TWB>$
by simp

moreover assume $\bigwedge Q'. Q \rightarrow (Calculus (STCalWB SWB TWB)) \ast Q' \Rightarrow Q' \downarrow<STCalWB SWB TWB>$

ultimately show $P \downarrow<STCalWB SWB TWB>$
by blast

qed

qed

lemma (in encoding-wrt-barbs) enc-and-TRel-impl-indRelRTPO-respects-success:
fixes success :: 'barbs
and TRel :: ('procT × 'procT) set
assumes encRS: enc-respects-barb-set {success}
and trelPS: rel-preserves-barb-set TRel TWB {success}
and trelRS: rel-reflects-barb-set TRel TWB {success}
shows rel-respects-barb-set (indRelRTPO TRel) (STCalWB SWB TWB) {success}

proof auto

fix $P \ Q$

assume $P \leq RT < TRel > Q$ and $P \downarrow<STCalWB SWB TWB>$

thus $Q \downarrow<STCalWB SWB TWB>$

proof induct

case (encR $S$
assume SourceTerm $S \downarrow<STCalWB SWB TWB>$

hence $S \downarrow<SWB>$

using STCalWB-hasBarbST
by blast

with encRS have $[S] \downarrow<TWB>$
by simp

thus TargetTerm $[[S]] \downarrow<STCalWB SWB TWB>$

using STCalWB-hasBarbST
by blast

next
case (source $S$
assume SourceTerm $S \downarrow<STCalWB SWB TWB>$
by simp

next
case (target $T1 \ T2$
assume $(T1, T2) \in TRel$

moreover assume TargetTerm $T1 \downarrow<STCalWB SWB TWB>$

hence $T1 \downarrow<TWB>$

using STCalWB-hasBarbST
by blast

ultimately have $T2 \downarrow<TWB>$

using trelPS
by simp

thus TargetTerm $T2 \downarrow<STCalWB SWB TWB>$

using STCalWB-hasBarbST
by blast

next
case (trans $P \ Q \ R'$)
assume $P \downarrow<STCalWB SWB TWB>$

and $P \downarrow<STCalWB SWB TWB> \Rightarrow Q \downarrow<STCalWB SWB TWB>$

and $Q \downarrow<STCalWB SWB TWB> \Rightarrow R \downarrow<STCalWB SWB TWB>$

thus $R \downarrow<STCalWB SWB TWB>$
by simp
qed

next
fix P Q
assume P \leq [\lbrack \llbracket T \rrbracket \rbrack < TRel \rangle Q and Q \downarrow < STCalWB SWB TWB > success
thus P \downarrow < STCalWB SWB TWB > success

proof induct
  case (encR S)
  assume TargetTerm ([S]) \downarrow < STCalWB SWB TWB > success
  hence [S] \downarrow < TWB > success
  using STCalWB-hasBarbST
  by blast
  with encRS have S \downarrow < SWB > success
  by simp
  thus SourceTerm S \downarrow < STCalWB SWB TWB > success
  using STCalWB-hasBarbST
  by blast

next
  case (source S)
  assume SourceTerm S \downarrow < STCalWB SWB TWB > success
  thus SourceTerm S \downarrow < STCalWB SWB TWB > success
  by simp

next
  case (target T1 T2)
  assume (T1, T2) \in TRel
  moreover assume TargetTerm T2 \downarrow < STCalWB SWB TWB > success
  hence T2 \downarrow < TWB > success
  using STCalWB-hasBarbST
  by blast
  ultimately have T1 \downarrow < TWB > success
  using trelRS
  by blast
  thus TargetTerm T1 \downarrow < STCalWB SWB TWB > success
  using STCalWB-hasBarbST
  by blast

next
  case (trans P Q R)
  assume R \downarrow < STCalWB SWB TWB > success
  and R \downarrow < STCalWB SWB TWB > success \Rightarrow Q \downarrow < STCalWB SWB TWB > success
  and Q \downarrow < STCalWB SWB TWB > success \Rightarrow P \downarrow < STCalWB SWB TWB > success
  thus P \downarrow < STCalWB SWB TWB > success
  by simp
qed

lemma (in encoding-wrt-barbs) enc-and-TRel-impl-indRelRTPO-respects-barbs:
  fixes TRel :: (procT \times procT) set
  assumes encRS: enc-respects-barbs
  and trelPS: rel-preserves-barbs TRel TWB
  and trelRS: rel-reflects-barbs TRel TWB
  shows rel-respects-barbs (indRelRTPO TRel) (STCalWB SWB TWB)
proof auto
  fix P Q x
  assume P \leq [\lbrack \llbracket T \rrbracket \rbrack < TRel \rangle Q and P \downarrow < STCalWB SWB TWB > x
  thus Q \downarrow < STCalWB SWB TWB > x
  proof induct
    case (encR S)
    assume SourceTerm S \downarrow < STCalWB SWB TWB > x
    hence S \downarrow < SWB > x
    using STCalWB-hasBarbST
    by blast
with $\text{encRS}$ have $\llbracket S \rrbracket \downarrow \llbracket \text{TWB} \rrbracket > x$
  by simp
thus $\text{TargetTerm} \; (\llbracket S \rrbracket \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  using $\text{STCalWB-hasBarbST}$
  by blast
next
case (source $S$)
assume $\text{SourceTerm} \; S \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
thus $\text{SourceTerm} \; S \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  by simp
next
case (target $T1 \; T2$)
assume $(T1, \; T2) \in \text{TRel}$
moreover assume $\text{TargetTerm} \; T1 \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  hence $T1 \downarrow \llbracket \text{TWB} \rrbracket > x$
    using $\text{STCalWB-hasBarbST}$
  by blast
ultimately have $T2 \downarrow \llbracket \text{TWB} \rrbracket > x$
    using $\text{trelPS}$
  by simp
thus $\text{TargetTerm} \; T2 \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  using $\text{STCalWB-hasBarbST}$
  by blast
next
case (trans $P \; Q \; R$)
assume $P \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  and $P \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x \Rightarrow Q \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  and $Q \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x \Rightarrow R \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
thus $R \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  by simp
qed
next
fix $P \; Q \; x$
assume $P \preceq \llbracket RT \rrbracket \llbracket \text{TRel} \rrbracket > Q$ and $Q \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
thus $P \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
proof induct
  case (encR $S$)
  assume $\text{TargetTerm} \; (\llbracket S \rrbracket \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
    hence $\llbracket S \rrbracket \downarrow \llbracket \text{TWB} \rrbracket > x$
      using $\text{STCalWB-hasBarbST}$
    by blast
with $\text{encRS}$ have $S \downarrow \llbracket \text{SWB} \rrbracket > x$
  by simp
thus $\text{SourceTerm} \; S \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  using $\text{STCalWB-hasBarbST}$
  by blast
next
case (source $S$)
assume $\text{SourceTerm} \; S \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
thus $\text{SourceTerm} \; S \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  by simp
next
case (target $T1 \; T2$)
assume $(T1, \; T2) \in \text{TRel}$
moreover assume $\text{TargetTerm} \; T2 \downarrow \llbracket \text{STCalWB SWB TWB} \rrbracket > x$
  hence $T2 \downarrow \llbracket \text{TWB} \rrbracket > x$
    using $\text{STCalWB-hasBarbST}$
  by blast
ultimately have $T1 \downarrow \llbracket \text{TWB} \rrbracket > x$
    using $\text{trelRS}$
  by blast

thus $\text{TargetTerm } T1 \downarrow STcalWB SWB TWB > x$

using $STcalWB\text{-hasBarbST}$

by blast

next

case (trans $P Q R$)

assume $R \downarrow STcalWB SWB TWB > x$

and $R \downarrow STcalWB SWB TWB > x \Rightarrow Q \downarrow STcalWB SWB TWB > x$

and $Q \downarrow STcalWB SWB TWB > x \Rightarrow P \downarrow STcalWB SWB TWB > x$

thus $P \downarrow STcalWB SWB TWB > x$

by simp

qed

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)** $\text{OC-SS-iff-source-target-rel:}$

**Fixes** success :: \"barbs

and $TRel$ :: \"(\\text{procT} \times \text{\textit{procT}})\" set

**Shows** (operational-corresponding ($TRel^*$))

\(|\begin{align*}
\land & \text{weak-reduction-bisimulation (} TRel^+ \text{) Target} \\
\land & \text{enc-weakly-respects-barb-set } \{\text{success}\} \\
\land & \text{rel-weakly-respects-barb-set } TRel \text{ TWB } \{\text{success}\}
\end{align*}\)

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

\(|\begin{align*}
\land & (\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{Rel}) \\
\land & (\forall T1 T2. (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{Rel} \longrightarrow (T1, T2) \in TRel^+) \\
\land & (\forall S T. (\text{SourceTerm S}, \text{TargetTerm T}) \in \text{Rel} \longrightarrow ([S], T) \in TRel^*) \\
\land & \text{weak-reduction-bisimulation Rel (STCal Source Target)} \\
\land & \text{rel-weakly-respects-barb-set Rel } (STCalWB SWB TWB) \{\text{success}\}
\end{align*}\)

**Proof** (rule iffI, (erule conjE)+)

**Assume** A1: rel-weakly-preserve-barb-set $TRel \text{ TWB } \{\text{success}\}$

and A2: rel-weakly-reflects-barb-set $TRel \text{ TWB } \{\text{success}\}$

and A3: enc-weakly-preserve-barb-set $\{\text{success}\}$

and A4: enc-weakly-reflects-barb-set $\{\text{success}\}$

**Def** rel: $\text{Rel} = \text{indRelRTPO-to-TRel}$

**Hence** B1: $\forall S. (\text{SourceTerm S}, \text{TargetTerm ([S]]) } \in \text{Rel}$

by (simp add: indRelRTPO-to-TRel)

**From** rel have B2: $\forall T1 T2. (T1, T2) \in TRel \longrightarrow (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{Rel}$

by (simp add: indRelRTPO-to-TRel)

**From** rel have B3: $\forall T1 T2. (\text{TargetTerm T1}, \text{TargetTerm T2}) \in \text{Rel} \longrightarrow (T1, T2) \in TRel^+$

by (simp add: indRelRTPO-to-TRel)

**From** rel have B4: $\forall S T. (\text{SourceTerm S}, \text{TargetTerm T}) \in \text{Rel} \longrightarrow ([S], T) \in TRel^*$

using indRelRTPO-to-TRel(2)[where $TRel=TRel$]

trans-closure-of-TRel-refl-cond[where $TRel=TRel$]

by simp

**Assume** operational-complete ($TRel^*$)

and operational-sound ($TRel^*$)

and weak-reduction-simulation ($TRel^+$) Target

and $\forall P Q. (P, Q) \in TRel^+ \land Q \longrightarrow Target^* Q'$

$\longrightarrow (\exists P'. P \longrightarrow Target^* P' \land (P', Q') \in TRel^+)$

**With** rel have B5: weak-reduction-bisimulation Rel (STCal Source Target)

using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where $TRel=TRel$]

by simp

from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) $\{\text{success}\}$

using enc-and-TRel-impl-indRelRTPO-weekly-respects-success[where $TRel=TRel$

and success=success]

by blast

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

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

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

245
\( \forall S, T. \ (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)
\( \wedge \text{weak-reduction-bisimulation Rel } (\text{STCal Source Target}) \)
\( \wedge \text{rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TBW}) \) \{success\}

apply (rule exI) using B1 B2 B4 B5 B6 by blast

next

assume \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \)
\( \wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+) \)
\( \wedge (\forall S, T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+) \)

from this obtain \( \text{Rel where } C1: \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \)
and \( C2: \forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \)
and \( C3: \forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+ \)
and \( C4: \forall S, T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)
and \( C5: \text{weak-reduction-bisimulation Rel } (\text{STCal Source Target}) \)
and \( C6: \text{rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TBW}) \) \{success\}

by auto

from \( C1 \ C2 \ C3 \ C4 \ C5 \) have \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}) \)
\( \wedge (\forall T1 T2. (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+ \)
\( \wedge (\forall S, T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+) \)
\( \wedge \text{weak-reduction-bisimulation Rel } (\text{STCal Source Target}) \)

by blast

hence \( \text{operational-corresponding } (\text{TRel}^+) \)
\( \wedge \text{weak-reduction-bisimulation } (\text{TRel}^+) \) \text{Target}

using \( \text{OC-iff-weak-reduction-bisimulation[where } TRel=\text{TRel}] \)
by auto

moreover have \( \exists \text{Rel. } (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge \text{rel-weakly-respects-barb-set Rel } (\text{STCalWB SWB TBW}) \) \{success\}

apply (rule exI) using \( C1 \ C6 \) by blast

hence \( \text{enc-weakly-respects-barb-set } \) \{success\}
using \( \text{success-sensitive-iff-source-target-rel-weakly-respects-success} \)
by auto

moreover have \( \text{rel-weakly-respects-barb-set } \text{TRel } \text{TBW} \) \{success\}

proof auto

fix \( TP TQ TP' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by simp

moreover assume \( TP \rightarrow (\text{Calculus } \text{TWB})^* \ TP' \ and \ TP' \downarrow \text{TWB} > \text{success} \)

hence \( \text{TargetTerm } TP' \downarrow <\text{STCalWB SWB TBW}> \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by blast

ultimately have \( \text{TargetTerm } TP' \downarrow <\text{STCalWB SWB TBW}> \text{success} \)
using \( C6 \)
by blast

thus \( TP' \downarrow <\text{TBW}> \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by blast

next

fix \( TP TQ TQ' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by simp

moreover assume \( TP \rightarrow (\text{Calculus } \text{TWB})^* \ TQ' \ and \ TQ' \downarrow <\text{TBW}> \text{success} \)

hence \( \text{TargetTerm } TQ' \downarrow <\text{STCalWB SWB TBW}> \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by blast

ultimately have \( \text{TargetTerm } TQ' \downarrow <\text{STCalWB SWB TBW}> \text{success} \)
using \( C6 \)

246
by blast
thus $TP = TWB > success$

using $STCalWB$-reachesBarbST
by blast
qed
ultimately show operational-corresponding ($TRel^*$)
\land weak-reduction-bisimulation ($TRel^+$) Target
\land enc-weakly-respects-barb-set \{success\} \land rel-weakly-respects-barb-set $TRel$ $TWB \ \{success\}$
by fast
qed

lemma (in encoding-wrt-barbs) OC-SS-RB-iff-source-rel:
fixes success :: 'bars
and $TRel$ :: ('procT \times 'procT) set
shows (operational-corresponding ($TRel^*$)
\land weak-reduction-bisimulation ($TRel^+$) Target
\land enc-weakly-respects-barbs \land enc-weakly-respects-barb-set \{success\}
\land rel-weakly-respects-barbs $TRel$ $TWB$ \land rel-weakly-respects-barb-set $TRel$ $TWB$ \{success\})
= ($\exists T. (\forall S. (\SourceTerm S, \TargetTerm ([S]))) \in Rel$
\land ($\forall T1, T2. (T1, T2) \in TRel \rightarrow (\TargetTerm T1, \TargetTerm T2) \in Rel$)
\land ($\forall T1, T2. (\TargetTerm T1, \TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+$
\land ($\forall S T. (\SourceTerm S, \TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^+$)
\land weak-bisimulation Rel ($STCal$ Source Target)
\land weakly-reflects-barbs $TRel$ ($STCalWB$ $SWB$ $TWB$)
\land weakly-reflects-barb-set Rel ($STCalWB$ $SWB$ $TWB$)
\land weakly-reflects-barb-set $TRel$ ($STCalWB$ $SWB$ $TWB$) \{success\})

proof (rule iffI, (erule conjE)+)

assume A1: rel-weakly-preserves-barb-set $TRel$ $TWB$ \{success\}
and A2: rel-weakly-reflects-barb-set $TRel$ $TWB$ \{success\}
and A3: enc-weakly-preserves-barb-set \{success\}
and A4: enc-weakly-reflects-barb-set \{success\}
and A5: rel-weakly-preserves-barbs $TRel$ $TWB$ and A6: rel-weakly-reflects-barbs $TRel$ $TWB$
and A7: enc-weakly-preserves-barbs and A8: enc-weakly-reflects-barbs

def rel: $\equiv indRelRTPO \ TRel$

hence B1: $\forall S. (\SourceTerm S, \TargetTerm ([S])) \in Rel$
by (simp add: indRelRTPO.encR)

from rel have B2: $\forall T1 T2. (T1, T2) \in TRel \rightarrow (\TargetTerm T1, \TargetTerm T2) \in Rel$
by (simp add: indRelRTPO.target)

from rel have B3: $\forall T1 T2. (\TargetTerm T1, \TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+$
by (simp add: indRelRTPO-to-TRel(4)[where $TRel=\ TRel$])

from rel have B4: $\forall S T. (\SourceTerm S, \TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^+$
using indRelRTPO-to-TRel(2)[where $TRel=\ TRel$
trans-closure-of-TRel-refl-cond[where $TRel=\ TRel$]

by simp

assume operational-complete ($TRel^*$)

and operational-sound ($TRel^*$)

and weak-reduction-simulation ($TRel^+$) Target

and $\forall P Q Q'. (P, Q) \in TRel^+ \land Q \rightarrow \TargetStar Q' \rightarrow (\exists P'. P \rightarrow \TargetStar P' \land (P', Q') \in TRel^+)$

with rel have B5: weak-reduction-bisimulation Rel ($STCal$ Source Target)
using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where $TRel=\ TRel$]

by simp

from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel ($STCalWB$ $SWB$ $TWB$) \{success\}
using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where $TRel=\ TRel$
and success=success]

by blast

from rel A5 A6 A7 A8 have B7: rel-weakly-respects-barbs Rel ($STCalWB$ $SWB$ $TWB$)
using enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs[where $TRel=\ TRel$

by blast

show $\exists \ TRel. (\forall S. (\SourceTerm S, \TargetTerm ([S])) \in Rel$
\land ($\forall T1 T2. (T1, T2) \in TRel \rightarrow (\TargetTerm T1, \TargetTerm T2) \in Rel$)
\land ($\forall T1 T2. (\TargetTerm T1, \TargetTerm T2) \in Rel \rightarrow (T1, T2) \in TRel^+$
\land ($\forall S T. (\SourceTerm S, \TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel^+$)
∀ 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}
apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast

next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —> (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —> (T1, T2) ∈ TRel+)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel —> ([S], T) ∈ TRel*)
∧ weak-reduction-bisimulation Rel (STCal Source Target)
∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
from this obtain Rel where C: (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —> (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —> (T1, T2) ∈ TRel+)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel —> ([S], T) ∈ TRel*)
∧ weak-reduction-bisimulation Rel (STCal Source Target)
∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
by auto

hence C1: (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
by simp
from C have C2: (∀ T1 T2. (T1, T2) ∈ TRel —> (TargetTerm T1, TargetTerm T2) ∈ Rel)
by simp
from C have C3: (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —> (T1, T2) ∈ TRel+)
by simp
from C have C4: (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel —> ([S], T) ∈ TRel*)
by simp
from C have C5: weak-reduction-bisimulation Rel (STCal Source Target)
by simp
from C have C7: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
apply (rule conjE) apply (erule conjE)+ by blast
from C have C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
apply (rule conjE) apply (erule conjE)+ by blast
from C1 C2 C3 C4 C5 have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel —> (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel —> (T1, T2) ∈ TRel+)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel —> ([S], T) ∈ TRel*)
∧ weak-reduction-bisimulation Rel (STCal Source Target)
by blast
hence operational-corresponding (TRel*)
∧ weak-reduction-bisimulation (TRel+) Target
using OC-iff-weak-reduction-bisimulation [where TRel=TRel]
by auto
moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
apply (rule exI) using C1 C6 by blast
hence enc-weakly-respects-barb-set {success}
using success-sensitive-iff-source-target-rel-weakly-respects-success
by auto
moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
apply (rule exI) using C1 C7 by blast
hence enc-weakly-respects-barbs
using enc-weakly-respects-barbs-iff-source-target-rel
by auto
moreover have rel-weakly-respects-barb-set TRel TWB {success}
proof auto
fix TP TQ TP’
assume (TP, TQ) ∈ TRel
with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel

248
by simp
moreover assume \( TP \mapsto (\text{Calculus TWB})^* \) \( TP' \) and \( TP' \downarrow <\text{TWB}> \) success
hence TargetTerm \( TP \downarrow <\text{STCalWB SWB TWB}> \) success
   using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TP \downarrow <\text{STCalWB SWB TWB}> \) success
   using \( C_6 \)
by blast
thus \( TP \downarrow <\text{TWB}> \) success
   using \( \text{STCalWB-reachesBarbST} \)
by blast
next
fix \( TP \) \( TQ \) \( TQ' \)
assume \((TP, TQ) \in TRel\)
with \( C_2 \) have \((\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}\)
by simp
moreover assume \( TP \mapsto (\text{Calculus TWB})^* \) \( TQ' \) and \( TQ' \downarrow <\text{TWB}> \) x
hence TargetTerm \( TQ \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TQ \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( C_7 \)
by blast
thus \( TP \downarrow <\text{TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
qed
moreover have \( \text{rel-weakly-respects-barbs} \) \( TRel \) \( TWB \)
proof auto
fix \( TP \) \( TQ \) \( x \) \( TP' \)
assume \((TP, TQ) \in TRel\)
with \( C_2 \) have \((\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}\)
by simp
moreover assume \( TP \mapsto (\text{Calculus TWB})^* \) \( TP' \) and \( TP' \downarrow <\text{TWB}> \) x
hence TargetTerm \( TP \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TQ \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( C_7 \)
by blast
thus \( TQ \downarrow <\text{TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
next
fix \( TP \) \( TQ \) \( x \) \( TQ' \)
assume \((TP, TQ) \in TRel\)
with \( C_2 \) have \((\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}\)
by simp
moreover assume \( TQ \mapsto (\text{Calculus TWB})^* \) \( TQ' \) and \( TQ' \downarrow <\text{TWB}> \) x
hence TargetTerm \( TQ \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TQ \downarrow <\text{STCalWB SWB TWB}> \) x
   using \( C_7 \)
by blast
thus \( TP \downarrow <\text{TWB}> \) x
   using \( \text{STCalWB-reachesBarbST} \)
by blast
qed
ultimately show operational-corresponding \( (TRel^*) \)
\land weak-reduction-bisimulation \( (TRel^*) \) \( Target \)

249
\[\land \text{enc-weakly-respects-barbs} \land \text{enc-weakly-respects-barb-set \{success\}}\]
\[\land \text{rel-weakly-respects-barbs TRel TWB} \land \text{rel-weakly-respects-barb-set TRel TWB \{success\}}\]
\[\text{by fast}\]
\[\text{qed}\]

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

- **fixes** success :: 'barbs
- and TRel :: ('procT x 'procT) set
- **shows** (operational-corresponding TRel \land preorder TRel \land weak-reduction-bisimulation TRel Target
\land \text{enc-weakly-respects-barb-set \{success\}}
\land \text{rel-weakly-respects-barb-set TRel TWB \{success\}}
\Rightarrow (\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})
\land \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
\land (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})
\land \text{weak-reduction-bisimulation \text{Rel} (STCal Source Target} \land preorder \text{Rel}
\land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}))

**proof** (rule iffI, (erule conjE)+)

- **assume** A1: rel-weakly-preserves-barb-set TRel TWB \{success\}
- and A2: rel-weakly-reflects-barb-set TRel TWB \{success\}
- and A3: enc-weakly-preserves-barb-set \{success\}
- and A4: enc-weakly-reflects-barb-set \{success\}
- and A5: preorder TRel

from A5 have A6: TRel' = TRel

- using trancl-id[of TRel] preorder-on-def

- by blast

from A5 have A7: TRel^+ = TRel

- using refl-trancl[of TRel] trancl-id[of TRel]

- unfolding refl-on-def preorder-on-def

- by auto

**def** rel: Rel\equiv indRelRTPO TRel

**hence** B1: \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}

- by (simp add: indRelRTPO.encR)

from rel A6 have B2: TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}

- using indRelRTPO-to-TRel(\_)[where TRel=TRel]

- by (auto simp add: indRelRTPO.target)

from rel A7 have B3: \forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}

- using indRelRTPO-to-TRel(\_)[where TRel=TRel]

- trans-closure-of.TRel-refl-cond[where TRel=TRel]

- by simp

**assume** operational-complete TRel and operational-sound TRel

- and weak-reduction-simulation TRel Target

- and \forall P Q Q'. (P, Q) \in TRel \land Q \rightarrow \text{Target} \rightarrow Q' \rightarrow (\exists P', P \rightarrow \text{Target} \rightarrow P' \land (P', Q') \in TRel)

**with** rel A6 A7 have B4: weak-reduction-bisimulation \text{Rel} (STCal Source Target)

- using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel=TRel]

- by simp

from rel A5 have B5: preorder Rel

- using indRelRTPO-is-preorder[where TRel=TRel]

- unfolding preorder-on-def

- by blast

from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}

- using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel]

- and success\equiv success]

- by blast

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

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

\lor (\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})

\lor \text{weak-reduction-bisimulation \text{Rel} (STCal Source Target} \land preorder \text{Rel}

\lor \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}}

**apply** (rule exI) using B1 B2 B3 B4 B5 B6 by blast

**next**

**assume** \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel})
\[ \begin{align*}
\wedge \ & \text{TRel} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\}\\
\wedge \ & (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})\\
\wedge \ & \text{weak-reduction-bisimulation } \text{Rel} (\text{STCal Source Target}) \wedge \text{preorder } \text{Rel}\\
\wedge \ & \text{rel-weakly-respects-barb-set } \text{Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}\end{align*} \]

from this obtain \text{Rel} where \( C_1 \): \((\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})\)
and \( C_2 \): \text{TRel} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\}
and \( C_3 \): \((\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel})\)
and \( C_4 \): weak-reduction-bisimulation \( \text{Rel} (\text{STCal Source Target}) \) and \( C_5 \): preorder \( \text{Rel} \)
and \( C_6 \): rel-weakly-respects-barb-set \( \text{Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}\) by \text{auto}

from \( C_1 C_2 C_3 C_4 C_5 \) have \( \exists \text{Rel}, (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge (\text{TRel} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\}) \)
\( \wedge (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}) \wedge \text{preorder } \text{Rel} \)
\( \wedge \text{rel-weakly-respects-barb-set } \text{Rel} (\text{STCal Source Target}) \)
by \text{blast}

hence operational-corresponding \( \text{TRel} \wedge \text{preorder } \text{TRel} \wedge \text{weak-reduction-bisimulation } \text{TRel} \text{Target} \)
using OC-wrt-preorder-iff-weak-reduction-bisimulation[where \text{TRel}=\text{TRel}]
by \text{simp}

moreover have \( \exists \text{Rel}, (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \wedge \text{rel-weakly-respects-barb-set } \text{Rel} (\text{STCalWB SWB TWB}) \{\text{success}\}\)
by \text{simp}

moreover have rel-weakly-respects-barb-set \( \text{TRel} \text{TWB} \{\text{success}\}\)
proof \text{auto}
fix \( TP \ TQ \ TP' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C_2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by \text{simp}
moreover assume \( TP \rightarrow (\text{Calculus TWB})^* \ (TP') \ \text{and} \ TP' \downarrow \text{TWB} > \text{success} \)
hence \( \text{TargetTerm } TP' \downarrow \text{STCalWB SWB TWB} > \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by \text{blast}
ultimately have \( \text{TargetTerm } TQ' \downarrow \text{STCalWB SWB TWB} > \text{success} \)
using \( \text{C6} \)
by \text{blast}
thus \( TQ' \downarrow \text{TWB} > \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by \text{blast}
next
fix \( TP \ TQ \ TQ' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C_2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by \text{simp}
moreover assume \( TQ \rightarrow (\text{Calculus TWB})^* \ (TQ') \ \text{and} \ TQ' \downarrow \text{TWB} > \text{success} \)
hence \( \text{TargetTerm } TQ' \downarrow \text{STCalWB SWB TWB} > \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by \text{blast}
ultimately have \( \text{TargetTerm } TP' \downarrow \text{STCalWB SWB TWB} > \text{success} \)
using \( \text{C6} \)
by \text{blast}
thus \( TP' \downarrow \text{TWB} > \text{success} \)
using \( \text{STCalWB-reachesBarbST} \)
by \text{blast}
qed
ultimately show operational-corresponding \( \text{TRel} \wedge \text{preorder } \text{TRel} \)
\wedge \text{weak-reduction-bisimulation } \text{TRel} \text{Target} \\
\wedge \text{enc-weakly-respects-barb-set } \{\text{success}\} \wedge \text{rel-weakly-respects-barb-set } \text{TRel} \text{TWB} \{\text{success}\}
by \text{fast}
qed
lemma (in encoding-wrt-barbs) OC-SS-RB-wrt preorder iff-source-target-rel:
  fixes success :: 'bars
  and TRel :: ('proc T × 'proc T) 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}
    ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
    ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
  ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success})
proof (rule iffI, (erule conjE)+)
  assume A1: rel-weakly-preserves-barbs TRel TWB and A2: rel-weakly-reflects-barbs TRel TWB
  and A3: enc-weakly-preserves-barbs and A4: enc-weakly-reflects-barbs
  and A5: preorder TRel
from A5 have A6: TRel⁺ = TRel
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
from A5 have A7: TRel⁺ = TRel
  using refl-trancl[of TRel] trancl-id[of TRel]
  unfolding preorder-on-def refl-on-def
  by auto
  def rel: Rel≡indRelRTPO TRel
  hence B1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  by (simp add: indRelRTPO.encR)
  from rel A6 have B2: TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
    using indRelRTPO-to-TRel(4)[where TRel= TRel]
    by (auto simp add: indRelRTPO.target)
  from rel A7 have B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
    using indRelRTPO-to-TRel(2)[where TRel= TRel]
    trans-closure-of-TRel-refl-cond[where TRel= TRel]
    by simp
  assume operational-complete TRel and operational-sound TRel
  and weak-reduction-simulation TRel Target
  and ∀ P Q Q'. (P, Q) ∈ TRel ∧ Q → Target* Q' → (∃ P'. P → Target* P' ∧ (P', Q') ∈ TRel)
  with rel A6 A7 have B4: weak-reduction-bisimulation Rel (STCal Source Target)
    using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel= TRel]
    by simp
  from rel A5 have B5: preorder Rel
  using indRelRTPO-is-preorder[where TRel= TRel]
  unfolding preorder-on-def
  by blast
  from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
    using enc-and-TRel-impl-indRelRTPO-weakly-respects-barbs[where TRel= TRel]
    by blast
  hence B7: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  by blast
  show ∃ Rel. ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
    ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
    ∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
    ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
    apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast
next
  assume ∃ Rel. ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
    ∧ TRel = {(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel}
∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}

from this obtain Rel where C1: (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
and C2: TRel = {((T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ S T. (SourceTerm S, TargetTerm (T)) ∈ Rel → ([S], T) ∈ TRel)
∧ weak-reduction-bisimulation Rel (STCal Source Target) and C5: preorder Rel
∧ C6: rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
by auto

moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (TRel = {((T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel})
∧ (∀ S T. (SourceTerm S, TargetTerm (T)) ∈ Rel → ([S], T) ∈ TRel) ∧ preorder Rel
∧ weak-reduction-bisimulation Rel (STCal Source Target)
by blast

hence operational-corresponding TRel ∧ preorder TRel ∧ weak-reduction-bisimulation TRel Target
using OC-wrt-preorder-iff-weak-reduction-bisimulation[where TRel=TRel]
by simp

moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
apply (rule exI) using C1 C6 by blast

hence enc-weakly-respects-barbs
using enc-weakly-respects-barbs-iff-source-target-rel
by simp

moreover hence enc-weakly-respects-barb-set {success}
by simp

moreover have rel-weakly-respects-barbs TRel TWB
proof auto
fix TP TQ x TP'
assume (TP, TQ) ∈ TRel
with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
by simp
moreover assume TP ↦−→ (Calculus TWB)* TP’ and TP’ ∈ TWB

hence TargetTerm TP' ∈ STCalWB SWB TWB
using STCalWB-reachesBarbST
by blast

ultimately have TargetTerm TQ ∈ STCalWB SWB TWB
using C6
by blast
thus TQ ∈ TWB
using STCalWB-reachesBarbST
by blast

next
fix TP TQ x TQ'
assume (TP, TQ) ∈ TRel
with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel
by simp
moreover assume TQ ↦−→ (Calculus TWB)* TQ’ and TQ’ ∈ TWB

hence TargetTerm TQ' ∈ STCalWB SWB TWB
using STCalWB-reachesBarbST
by blast

ultimately have TargetTerm TP' ∈ STCalWB SWB TWB
using C6
by blast
thus TP' ∈ TWB
using STCalWB-reachesBarbST
by blast

qed
moreover hence rel-weakly-respects-barb-set TRel TWB {success}
by blast
ultimately show operational-corresponding TRel ∧ preorder TRel
\(\land\) weak-reduction-bisimulation \(T_{rel}\) Target
\(\land\) enc-weakly-respects-barbs \(\land\) rel-weakly-respects-barbs \(T_{rel}\) TWB
\(\land\) enc-weakly-respects-barb-set \{success\} \(\land\) rel-weakly-respects-barb-set \(T_{rel}\) TWB \{success\}
by fast
qed

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

**Lemma** (in encoding-wrt-barbs) WOC-SS-wrt-preorder-iff-source-target-rel:
- **fixes** success :: 'bars
- **and** \(T_{rel}\) :: \((\text{proc} T \times \text{proc} T)\) set
- **shows** (weakly-operational-coringresponding \(T_{rel}\) \(\land\) preorder \(T_{rel}\)
  \(\land\) weak-reduction-corrrespondence-simulation \(T_{rel}\) Target
  \(\land\) enc-weakly-respects-barb-set \{success\}
  \(\land\) rel-weakly-respects-barb-set \(T_{rel}\) TWB \{success\})
  = (\(\exists T_{rel}\). \((\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}\)
  \(\land\) \(T_{rel}\) = \{((T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2)) \in \text{Rel}\}
  \(\land\) (\(\forall S T. \text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in T_{rel}\)
  \(\land\) weak-reduction-corrrespondence-simulation Rel (\(ST_{Cal}\ Source\ Target\) \(\land\) preorder \(T_{rel}\)
  \(\land\) rel-weakly-respects-barb-set Rel (\(ST_{Cal}WB\ SWB\ TWB\) \{success\}))

**Proof** (rule iffI, (erule conjE)+)
- **assume** \(A_1\): rel-weakly-preserves-barb-set \(T_{rel}\) TWB \{success\}
- **and** \(A_2\): rel-weakly-reflects-barb-set \(T_{rel}\) TWB \{success\}
- **and** \(A_3\): enc-weakly-preserves-barb-set \{success\}
- **and** \(A_4\): enc-weakly-reflects-barb-set \{success\}
- **and** \(A_5\): preorder \(T_{rel}\)

from \(A_5\) have \(A_6\): \(T_{rel}^{+} = T_{rel}\)
using trancl-id[of \(T_{rel}\)]

**unfolding** preorder-on-def
by blast

from \(A_5\) \(A_6\) have \(A_7\): \(T_{rel}^{*} = T_{rel}\)
using refl-transl[of \(T_{rel}\)] trancl-id[of \(T_{rel}\)]

**unfolding** preorder-on-def refl-on-def
by auto

**def** rel: \(\text{Rel} = \text{indRelRTPO} \ T_{rel}\)

**hence** \(B_1\): \(\forall S. \text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}\)
by (simp add: indRelRTPO.encR)

from \(\text{rel}\) \(A_6\) have \(B_2\): \(T_{rel} = \{((T_1, T_2), (\text{TargetTerm} T_1, \text{TargetTerm} T_2)) \in \text{Rel}\}
using indRelRTPO-to-TRel(\(\eta\)) [where \(T_{rel} = T_{rel}\)]
by (auto simp add: indRelRTPO.target)

from \(\text{rel}\) \(A_7\) have \(B_3\): \(\forall S T. \text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in T_{rel}\)
using indRelRTPO-to-TRel(\(\eta\)) [where \(T_{rel} = T_{rel}\)]
simp closure-of-TRel-refl-cond [where \(T_{rel} = T_{rel}\)]

by simp

**assume** operational-complete \(T_{rel}\) and weakly-operational-sound \(T_{rel}\)

**and** weak-reduction-simulation \(T_{rel}\) Target

**and** \(\forall P Q Q' . (P, Q) \in \text{Rel} \land Q \rightarrow T_{rel} \rightarrow \text{Target} \ast Q'\)

\(\rightarrow \exists P'' Q''. (P', Q'') \rightarrow \text{Target} \ast Q'\ast (P'', Q'') \in T_{rel}\)

with \(\text{rel}\) \(A_6\) \(A_7\) have \(B_4\): weak-reduction-correspondence-simulation Rel (\(ST_{Cal}\ Source\ Target\)
using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation [where \(T_{rel} = T_{rel}\)]
by simp

from \(\text{rel}\) \(A_5\) have \(B_5\): preorder Rel

using indRelRTPO-is-preorder [where \(T_{rel} = T_{rel}\)]

**unfolding** preorder-on-def
by blast

from \(\text{rel}\) \(A_1\) \(A_2\) \(A_3\) \(A_4\) have \(B_6\): rel-weakly-respects-barb-set Rel (\(ST_{Cal}WB\ SWB\ TWB\) \{success\}
using enc-and-TRel-impl-indRelRTPO-weakly-respects-success [where \(T_{rel} = T_{rel} \land\)

and success = success]
by blast

show \exists Rel (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
∧ TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel\}
∧ (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel
∧ rel-weekly-respects-barb-set Rel (STCalWB SWB TBW) \{ success \}
apply (rule exI) using B1 B2 B3 B4 B5 B6 by blast

next
assume \exists Rel (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
∧ TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel\}
∧ (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel
∧ rel-weekly-respects-barb-set Rel (STCalWB SWB TBW) \{ success \}
from this obtain Rel where C1: (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
and C2: TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) \in Rel\}
and C3: (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel)
and C4: weak-reduction-correspondence-simulation Rel (STCal Source Target)
and C5: preorder Rel and C6: rel-weekly-respects-barb-set Rel (STCalWB SWB TBW) \{ success \}
by auto
from C1 C2 C3 C4 C5 have \exists Rel (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
∧ (\forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel)
∧ weak-reduction-correspondence-simulation Rel (STCal Source Target)
by blast
hence weekly-operational-corresponding TRel \land preorder TRel
∧ weak-reduction-correspondence-simulation TRel Target
using WOC-wrt-preorder-if-reduction-correspondence-simulation[where TRel=TRel]
by simp
moreover have \exists Rel (\forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
∧ rel-weekly-respects-barb-set Rel (STCalWB SWB TBW) \{ success \}
apply (rule exI) using C1 C6 by blast
hence enc-weekly-respects-barb-set \{ success \}
using success-sensitive-if-source-target-rel-weekly-respects-success
by simp
moreover have rel-weekly-respects-barb-set TRel TBW \{ success \}
proof auto
fix TP TQ TP'
assume (TP, TQ) \in TRel
with C2 have (TargetTerm TP, TargetTerm TQ) \in Rel
by simp
moreover assume TP \rightarrow (Calculus TWB)* TP' \and TP'↓\prec\prec TBW > success
hence TargetTerm TP'↓\prec\prec STCalWB SWB TBW > success
using STCalWB-reachesBarbST
by blast
ultimately have TargetTerm TQ'\prec\prec STCalWB SWB TBW > success
using C6
by blast
thus TQ'\prec\prec TBW > success
using STCalWB-reachesBarbST
by blast
next
fix TP TQ TQ'
assume (TP, TQ) \in TRel
with C2 have (TargetTerm TP, TargetTerm TQ) \in Rel
by simp
moreover assume TQ \rightarrow (Calculus TWB)* TQ' \and TQ'\prec\prec TBW > success
hence TargetTerm TQ\prec\prec STCalWB SWB TBW > success
using STCalWB-reachesBarbST
by blast
ultimately have TargetTerm TP\prec\prec STCalWB SWB TBW > success
using C6
by blast
thus $TP^* \triangleleft TWB$ \text{success}
using $STCalWB$-reachesBarb$ST$
by blast

qed

ultimately show weakly-operational-corresponding $TRel$ \land preorder $TRel$
\land weak-reduction-correspondence-simulation $TRel$ Target
\land enc-weakly-respects-barb-set \{ success \} \land rel-weakly-respects-barb-set $TRel$ TWB \{ success \}
by fast

qed

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

fixes success :: 'barbs
and $TRel$ :: ('procT \times 'procT) set

shows (weakly-operational-corresponding $TRel$ \land preorder $TRel$
\land weak-reduction-correspondence-simulation $TRel$ Target
\land enc-weakly-respects-barbs \land enc-weakly-respects-barb-set \{ success \}
\land rel-weakly-respects-barbs $TRel$ TWB \land rel-weakly-respects-barb-set $TRel$ TWB \{ success \})
= (\exists Rel. \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel)
\land TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}
\land \forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel
\land weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel
\land rel-weakly-respects-barbs Rel (STCalWB SWB TWB)
\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{ success \})

proof (rule iffI, (erule conjI)+)

assume A1: rel-weakly-preserves-barb-set $TRel$ TWB \{ success \}
and A2: rel-weakly-reflects-barb-set $TRel$ TWB \{ success \}
and A3: enc-weakly-preserves-barb-set \{ success \}
and A4: enc-weakly-reflects-barb-set \{ success \}
and A5: preorder $TRel$

and A1': rel-weakly-preserves-barbs $TRel$ TWB \land A2': rel-weakly-reflects-barbs $TRel$ TWB
and A3': enc-weakly-preserves-barbs and A4': enc-weakly-reflects-barbs

from A5 have A6: $TRel^+ = TRel$
using trancl-id[of $TRel$]

unfolding preorder-on-def

by blast

from A5 A6 have A7: $TRel^* = TRel$
using reflcl-trancl[of $TRel$] trancl-id[of $TRel$]

unfolding preorder-on-def refl-on-def

by auto
def rel: Rel\equiv indRelRTPO $TRel$

hence B1: \forall S. (SourceTerm S, TargetTerm ([S])) \in Rel
by (simp add: indRelRTPO.encR)

from rel A6 have B2: $TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) \in Rel\}$
using indRelRTPO-to-TRel(4)[where $TRel=TRel$]
by (auto simp add: indRelRTPO.target)

from rel A7 have B3: \forall S T. (SourceTerm S, TargetTerm T) \in Rel \rightarrow ([S], T) \in TRel
using indRelRTPO-to-TRel(2)[where $TRel=TRel$]
trans-closure-of-TRel-refl-cond[where $TRel=TRel$]
by simp

assume operational-complete $TRel$ and weakly-operational-sound $TRel$
and weak-reduction-simulation $TRel$ Target

and \forall P Q Q'. (P, Q) \in TRel \land Q \rightarrow Target\* Q' \rightarrow
Exists P'' Q''. P \rightarrow Target\* P'' \land Q' \rightarrow Target\* Q'' \land (P'', Q'') \in TRel

with rel A6 A7 have B4: weak-reduction-correspondence-simulation Rel (STCal Source Target)
using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where $TRel=TRel$]
by simp

from rel A5 have B5: preorder $Rel$

using indRelRTPO-is-preorder[where $TRel=TRel$]

unfolding preorder-on-def

by blast

256
from rel $A_1$ $A_2$ $A_3$ $A_4$ have $B_6$: rel-weakly-respects-barb-set $R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) {success} 
using enc-and-$T_{\text{Rel}}$-impl-indRelRTPO-weakly-respects-success [where $T_{\text{Rel}}$=$T_{\text{Rel}}$]
and success=success
by blast
from rel $A_1'$ $A_2'$ $A_3'$ $A_4'$ have $B_7$: rel-weakly-respects-barbs $R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
using enc-and-$T_{\text{Rel}}$-impl-indRelRTPO-weakly-respects-barbs [where $T_{\text{Rel}}$=$T_{\text{Rel}}$]
by blast
show $\exists R_{\text{Rel}}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in R_{\text{Rel}}) 
\land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in R_{\text{Rel}}\} 
\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in R_{\text{Rel}} \implies ([S], T) \in T_{\text{Rel}}) 
\land weak\text{-}reduction\text{-}correspondence\text{-}simulation\text{-}rel\text{-}\text{source}\text{-}target\text{-}rel (ST\text{Cal Source Target}) 
\land preorder\text{-}rel \land rel\text{-}weakly\text{-}respects\text{-}barbs R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
\land rel\text{-}weakly\text{-}respects\text{-}barb\text{-}set $R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) {success} 
apply (rule exI) using $B_1$ $B_2$ $B_3$ $B_4$ $B_5$ $B_6$ $B_7$ by blast
next
assume $\exists R_{\text{Rel}}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in R_{\text{Rel}}) 
\land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in R_{\text{Rel}}\} 
\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in R_{\text{Rel}} \implies ([S], T) \in T_{\text{Rel}}) 
\land weak\text{-}reduction\text{-}correspondence\text{-}simulation\text{-}rel (ST\text{Cal Source Target}) 
\land preorder\text{-}rel \land rel\text{-}weakly\text{-}respects\text{-}barbs R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
\land rel\text{-}weakly\text{-}respects\text{-}barb\text{-}set $R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) {success} 
from this obtain $R_{\text{Rel}}$ where $C_1$: $(\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in R_{\text{Rel}}) 
\land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in R_{\text{Rel}}\} 
\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in R_{\text{Rel}} \implies ([S], T) \in T_{\text{Rel}}) 
\land weak\text{-}reduction\text{-}correspondence\text{-}simulation\text{-}rel (ST\text{Cal Source Target}) 
\land preorder\text{-}rel and $C_7$: rel-weakly-respects-barbs $R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
by auto
from $C_1$ $C_2$ $C_3$ $C_4$ $C_5$ have $\exists R_{\text{Rel}}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in R_{\text{Rel}}) 
\land (T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in R_{\text{Rel}}\} 
\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in R_{\text{Rel}} \implies ([S], T) \in T_{\text{Rel}}) 
\land weak\text{-}reduction\text{-}correspondence\text{-}simulation\text{-}rel (ST\text{Cal Source Target}) 
\land preorder\text{-}rel \land rel\text{-}weakly\text{-}respects\text{-}barbs R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
by auto
hence weakly\text{-}operational\text{-}corresponding $T_{\text{Rel}} \land preorder\text{-}T_{\text{Rel}} 
\land weak\text{-}reduction\text{-}correspondence\text{-}simulation \land Target\text{-}rel 
using WOC\text{-}wrt\text{-}preorder\text{-}iff\text{-}reduction\text{-}correspondence\text{-}simulation [where $T_{\text{Rel}}$=$T_{\text{Rel}}$]
by simp
moreover have $\exists R_{\text{Rel}}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in R_{\text{Rel}}) 
\land rel\text{-}weakly\text{-}respects\text{-}barbs R_{\text{Rel}}$ ($ST\text{Cal}WB$ $SWB$ $TWB$) 
apply (rule exI) using $C_1$ $C_7$ by blast
hence enc\text{-}weakly\text{-}respects\text{-}barbs 
using enc\text{-}weakly\text{-}respects\text{-}barbs\text{-}iff\text{-}source\text{-}target\text{-}rel 
by simp
moreover hence enc\text{-}weakly\text{-}respects\text{-}barb\text{-}set {success} 
by simp
moreover have rel\text{-}weakly\text{-}respects\text{-}barbs $T_{\text{Rel}}$ TWB
proof auto
fix $TP$ $TQ$ $x$ $TP' $
assume $(TP, TQ) \in T_{\text{Rel}}$
with $C_2$ have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in R_{\text{Rel}}$
by simp
moreover assume $TP \rightarrow (\text{Calculus TWB})* TP' \land TP' \downarrow <\text{TWB}>x$
hence $\text{TargetTerm } TP \downarrow <\text{STCal} WB\text{ SWB} \text{ TWB}>x$
using $ST\text{Cal}WB\text{-}reaches\text{BarbST}$
by blast
ultimately have $\text{TargetTerm } TQ \downarrow <\text{STCal} WB\text{ SWB} \text{ TWB}>x$
using $C_7$
by blast
thus $TQ \downarrow <\text{TWB}>x$
using $ST\text{Cal}WB\text{-}reaches\text{BarbST}$
by blast
next
lemma (in encoding-wrt-barbs) SOC-SS-wrt-preorder-iff-source-target-rel:

fixes success :: 'barbs
and TRel :: ('procT × 'procT) set
shows (strongly-operational-corresponding TRel ∧ preorder TRel
∧ strong-reduction-bisimulation TRel Target
∧ enc-respects-barb-set {success} ∧ rel-respects-barb-set TRel TWB {success})
= (∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success})

proof (rule iffI, (erule conjE)+)

assume A1: rel-preserves-barb-set TRel TWB {success}
and A2: rel-reflects-barb-set TRel TWB {success}
and A3: enc-preserves-barb-set {success} and A4: enc-reflects-barb-set {success}
and A5: preorder TRel
from A5 have A6: TRel+ = TRel
using trancl-id[of TRel]
unfolding preorder-on-def
by blast
from A5 A6 have A7: TRel' = TRel
using reflcl-trancl[of TRel] trancl-id[of TRel]
unfolding preorder-on-def refl-on-def
by auto
def rel:: Rel⇒indRelRTPO TRel
hence B1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
by (simp add: indRelRTPO.encR)
from rel A6 have B2: TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel}
using indRelRTPO-to-TRel(4)[where TRel=TRel]
by (auto simp add: indRelRTPO.target)
from rel A7 have B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
using indRelRTPO-to-TRel(2)[where TRel=TRel]
trans-closure-of-TRel-refl-cond[where TRel=TRel]

by simp

assume strongly-operational-complete TRel and strongly-operational-sound TRel

and strong-reduction-simulation TRel Target

and ∀ P Q Q′, (P, Q) ∈ TRel ∧ Q ⇒ Target Q′ ⇒ (∃ P′, P ⇒ Target P′∧ (P′, Q′) ∈ TRel)

with rel A6 A7 have B4: strong-reduction-bisimulation Rel (STCal Source Target)

using SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]

by simp

from rel A5 have B5: preorder Rel

using indRelRTPO-is-preorder[where TRel=TRel]

unfolding preorder-on-def

by blast

from rel A1 A2 A3 A4 have B6: rel-respects-barb-set Rel (STCalWB SWB TWB) {success}

using enc-and-TRel-impl-indRelRTPO-respects-success[where TRel=TRel and success=success]

by blast

show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel\}

∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⇒ ([S], T) ∈ TRel)

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

∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success}

apply (rule exI) using B1 B2 B3 B4 B5 B6 by blast

next

assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel\}

∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⇒ ([S], T) ∈ TRel)

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

∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success}

from this obtain Rel where C1: (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

and C2: TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel\}

and C3: (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⇒ ([S], T) ∈ TRel)

and C4: strong-reduction-bisimulation Rel (STCal Source Target) and C5: preorder Rel

and C6: rel-respects-barb-set Rel (STCalWB SWB TWB) {success}

by auto

from C1 C2 C3 C4 C5 have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ (TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel\})

∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel ⇒ ([S], T) ∈ TRel) ∧ preorder Rel

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

by blast

hence strongly-operational-corresponding TRel ∧ preorder TRel

∧ strong-reduction-bisimulation TRel Target

using SOC-wrt-preorder-iff-strong-reduction-bisimulation[where TRel=TRel]

by simp

moreover have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)

∧ rel-respects-barb-set Rel (STCalWB SWB TWB) {success}

apply (rule exI) using C1 C6 by blast

hence enc-respects-barb-set {success}

using success-sensitive-iff-source-target-rel-respects-success

by simp

moreover have rel-respects-barb-set TRel TWB {success}

proof auto

fix TP TQ

assume (TP, TQ) ∈ TRel

with C2 have (TargetTerm TP, TargetTerm TQ) ∈ Rel

by simp

moreover assume TP↓<TWB>success

hence TargetTerm TP↓<STCalWB SWB TWB>success

using STCalWB-hasBarbST

by blast

ultimately have TargetTerm TQ↓<STCalWB SWB TWB>success

using C6

by blast
thus $TQ \downarrow <TWB>\text{success}$
  using $STCalWB\text{-hasBarbST}$
  by blast

next

fix $TP \ TQ$
assume $(TP, TQ) \in TRel$
with $C2$ have $(\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel}$
  by simp
moreover assume $TQ \downarrow <TWB>\text{success}$

hence $\text{TargetTerm } TQ \downarrow <STCalWB \swb TWB>\text{success}$
  using $STCalWB\text{-hasBarbST}$
  by blast

ultimately have $\text{TargetTerm } TP \downarrow <STCalWB \swb TWB>\text{success}$
  using $C6$
  by blast

thus $TP \downarrow <TWB>\text{success}$
  using $STCalWB\text{-hasBarbST}$
  by blast

qed

ultimately show strongly-operational-corresponding $TRel \land \text{preorder } TRel$

$\land \text{enc-respects-barbs } TRel \text{ TWB}$

$\land \text{rel-respects-barbs } TRel \text{ TWB } \{\text{success}\}$

by fast

qed

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

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

shows (strongly-operational-corresponding $TRel \land \text{preorder } TRel$

$\land \text{enc-respects-barbs } TRel \text{ TWB}$

$\land \text{enc-respects-barbs } TRel \text{ TWB } \{\text{success}\}$

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

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

$\land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in TRel)$

$\land \text{strong-reduction-bisimulation } \text{Rel } (STCal \text{ Source } \text{Target}) \land \text{preorder } \text{Rel}$

$\land \text{enc-respects-barbs } \text{Rel } (STCalWB \swb SWB \swb TWB)$

$\land \text{rel-respects-barbs } \text{Rel } (STCalWB \swb SWB \swb TWB ) \{\text{success}\}$

$\land \text{rel-respects-barb-set } \text{Rel } (STCalWB \swb SWB \swb TWB ) \{\text{success}\}$)

proof (rule iffI, (erule conjE)+)

assume $A1$: rel-preserves-barbs $TRel \text{ TWB}$ and $A2$: rel-reflects-barbs $TRel \text{ TWB}$
and $A3$: enc-preserves-barbs and $A4$: enc-reflects-barbs
and $A5$: preorder $TRel$

from $A5$ have $A6$: $TRel^+ = TRel$
  using trancl-id[of $TRel$]

unfolding preorder-on-def
by blast

from $A5$ have $A7$: $TRel^* = TRel$
  using refl-trancl[of $TRel$] trancl-id[of $TRel$]

unfolding preorder-on-def refl-on-def
by auto

def rel: $\text{Rel} \equiv \text{indRelRTPO } TRel$

hence $B1$: $\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}$
  by (simp add: indRelRTPO_encR)

from rel $A6$ have $B2$: $TRel = \{(T1, T2), (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\}$
  using indRelRTPO-to-TRel(4)[where $TRel=TRel$]
  by (auto simp add: indRelRTPO_target)

from rel $A7$ have $B3$: $\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in TRel$
  using indRelRTPO-to-TRel(2)[where $TRel=TRel$]
  trans-closure-of-TRel-refl-cond[where $TRel=TRel$]
  by simp

assume strongly-operational-complete $TRel$ and strongly-operational-sound $TRel$
and strong-reduction-simulation \( T_{\text{Rel}} \) \( \text{Target} \)
and \( \forall P Q Q'. (P, Q) \in T_{\text{Rel}} \land Q \rightarrow \text{Target} Q' \rightarrow (\exists P'. P \rightarrow \text{Target} P' \land (P', Q') \in T_{\text{Rel}}) \)

with \( \text{rel A6 A7 have B4: strong-reduction-bisimulation Rel (STCal Source Target)} \)
using SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where \( T_{\text{Rel}} = T_{\text{Rel}} \)]
by simp
from \( \text{rel A5 have B5: preorder Rel} \)
using indRelRTPO-is-preorder[where \( T_{\text{Rel}} = T_{\text{Rel}} \)]
unfolding preorder-on-def
by blast
from \( \text{rel A1 A2 A3 A4 have B6: rel-respects-barbs Rel (STCalWB SWB TWB)} \)
using enc-and-TRel-impl-indRelRTPO-respects-barbs[where \( T_{\text{Rel}} = T_{\text{Rel}} \)]
by blast
hence \( B7: \text{rel-respects-barb-set Rel (STCalWB SWB TWB} \) \{ \text{success} \}
by blast
show \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\} \)
\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T_{\text{Rel}}) \)
\( \land \text{strong-reduction-bisimulation Rel (STCal Source Target) } \land \text{preorder Rel} \)
\( \land \text{rel-respects-barbs Rel (STCalWB SWB TWB)} \)
\( \land \text{rel-respects-barb-set Rel (STCalWB SWB TWB) } \{ \text{success} \} \)
apply (rule exI) using \( B1 B2 B3 B4 B5 B6 \) by blast
next
assume \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \land T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\} \)
\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T_{\text{Rel}}) \)
\( \land \text{strong-reduction-bisimulation Rel (STCal Source Target) } \land \text{preorder Rel} \)
\( \land \text{rel-respects-barbs Rel (STCalWB SWB TWB)} \)
\( \land \text{rel-respects-barb-set Rel (STCalWB SWB TWB) } \{ \text{success} \} \)
from this obtain \( \text{Rel where C1: (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})} \)
and \( C2: T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\} \)
and \( C3: (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T_{\text{Rel}}) \land \text{preorder Rel} \)
and \( C4: \text{strong-reduction-bisimulation Rel (STCal Source Target)} \) and \( C5: \text{preorder Rel} \)
and \( C6: \text{rel-respects-barbs Rel (STCalWB SWB TWB)} \)
by auto
from \( C1 C2 C3 C4 C5 \) have \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \land (T_{\text{Rel}} = \{(T_1, T_2), (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}\}) \)
\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in T_{\text{Rel}}) \land \text{preorder Rel} \)
\( \land \text{strong-reduction-bisimulation Rel (STCal Source Target)} \)
by blast
hence strongly-operational-corresponding \( T_{\text{Rel}} \) \( \land \text{preorder } T_{\text{Rel}} \)
using SOC-wrt-preorder-iff-strong-reduction-bisimulation[where \( T_{\text{Rel}} = T_{\text{Rel}} \)]
by simp
moreover have \( \exists \text{Rel}. (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \land \text{rel-respects-barbs Rel (STCalWB SWB TWB)} \)
apply (rule exI) using \( C1 C6 \) by blast
hence \( \text{enc-respects-barbs} \)
using \( \text{enc-respects-barbs-iff-source-target-rel} \)
by simp
moreover hence \( \text{enc-respects-barb-set } \{ \text{success} \} \)
by simp
moreover have \( \text{rel-respects-barbs } T_{\text{Rel}} \) \( \text{TWB} \)
proof auto
fix \( TP TQ x \)
assume \( (TP, TQ) \in T_{\text{Rel}} \)
with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by simp
moreover assume \( TP \downarrow_{\text{TWB}} x \)
then \( \text{TargetTerm } TP \downarrow_{\text{STCalWB SWB TWB}} x \)
using \( \text{STCalWB-hasBarbST} \)
by blast
ultimately have TargetTerm $TQ \downarrow<STCalWB \ SWB \ TWB>x$
   using $C6$
   by blast
thus $TQ\downarrow<TWB>x$
   using $STCalWB-hasBarbST$
   by blast

next
fix $TP \ TQ \ x$
assume $(TP, TQ) \in TRel$
with $C2$ have $(TargetTerm \ TP, TargetTerm \ TQ) \in Rel$
   by simp
moreover assume $TQ\downarrow<TWB>x$
hence TargetTerm $TQ\downarrow<STCalWB \ SWB \ TWB>x$
   using $STCalWB-hasBarbST$
   by blast
ultimately have TargetTerm $TP\downarrow<STCalWB \ SWB \ TWB>x$
   using $C6$
   by blast
thus $TP\downarrow<TWB>x$
   using $STCalWB-hasBarbST$
   by blast
qed

moreover hence rel-respects-barb-set $TRel \ TWB \ \{success\}$
   by blast
ultimately show strongly-operational-corresponding $TRel \ Target$
   \& preorder $TRel$
   \& strong-reduction-bisimulation $TRel \ Target$
   \& enc-respects-barbs \& rel-respects-barbs $TRel \ TWB$
   \& enc-respects-barb-set $\{success\}$ \& rel-respects-barb-set $TRel \ TWB \ \{success\}$
   by fast
qed

Next we also add divergence reflection to operational correspondence and success sensitiveness.

lemma (in encoding) enc-and-TRelimpl-indRelRTPO-reflect-divergence:
   fixes $TRel : (\text{\textquoteleft}procT \times \text{\textquoteleft}procT\text{\textquoteright}) \ \text{set}$
   assumes encRD: enc-reflects-divergence
   and trelRD: rel-reflects-divergence $TRel \ Target$
   shows rel-reflects-divergence (indRelRTPO $TRel$) ($STCal \ Source \ Target$)
proof auto
   fix $P \ Q$
   assume $P \leq_{\{\text{\textquoteleft}\text{\textquoteleft}RT<\text{\textquoteleft}TRel\text{\textquoteright}\text{\textquoteright}\}Q}$ and $Q \rightarrow(STCal \ Source \ Target)\omega$
   thus $P \rightarrow(STCal \ Source \ Target)\omega$
   by (simp add: STCal-divergent (1))
   next
   case (source $S$)
   assume $SourceTerm \ S \rightarrow(STCal \ Source \ Target)\omega$
   thus $SourceTerm \ S \rightarrow(STCal \ Source \ Target)\omega$
   by simp
   next
   case (target $T1 \ T2$)
   assume $(T1, T2) \in TRel$
   moreover assume $TargetTerm \ T2 \rightarrow(STCal \ Source \ Target)\omega$
   hence $T2 \rightarrow(Target)\omega$
   by (simp add: STCal-divergent (2))
ultimately have \( T_1 \rightarrow (\text{Target}) \omega \)

using \( \text{treRD} \)

by \( \text{blast} \)

thus \( \text{TargetTerm} \ T_1 \rightarrow (\text{STCal Source Target}) \omega \)

by \( (\text{simp add: STCal-divergent(2)}) \)

next

case (\( \text{trans P Q R} \))

assume \( R \rightarrow (\text{STCal Source Target}) \omega \)

and \( R \rightarrow (\text{STCal Source Target}) \omega \rightarrow Q \rightarrow (\text{STCal Source Target}) \omega \)

and \( Q \rightarrow (\text{STCal Source Target}) \omega \rightarrow P \rightarrow (\text{STCal Source Target}) \omega \)

thus \( P \rightarrow (\text{STCal Source Target}) \omega \)

by \( \text{simp} \)

qed

qed

\begin{lemma} \text{\textbf{encoding-wrt-barbs}} \text{\textbf{OC-SS-DR-iff-source-target-rel}}: \end{lemma}

\begin{itemize}
  \item \textbf{fixes} \( \text{success : 'barbs} \)
  \item \textbf{and} \( \text{TRel} \) \( \rightarrow \) \( \text{('procT} \times \text{'procT}) \text{ set} \)
  \item \textbf{shows} (\text{operational-corr} \text{(TRel\( ^+ \))})
    \begin{itemize}
      \item weak-reduction-bisimulation \( \text{(TRel\( ^+ \)) Target} \)
      \item \( \text{enc-weakly-respects-barb-set \{success\}} \)
      \item \( \text{rel-weakly-respects-barb-set TRel TWB \{success\}} \)
      \item \( \text{enc-reflects-divergence \& rel-reflects-divergence TRel Target} \)
    \end{itemize}
\end{itemize}

\begin{proof}
\begin{itemize}
  \item assume \( A1\): \( \text{rel-weakly-preserves-barb-set TRel TWB \{success\}} \)
  \item and \( A2\): \( \text{rel-weakly-reflects-barb-set TRel TWB \{success\}} \)
  \item and \( A3\): \( \text{enc-weakly-preserves-barb-set \{success\}} \)
  \item and \( A4\): \( \text{enc-weakly-reflects-barb-set \{success\}} \)
  \item and \( A5\): \( \text{rel-reflects-divergence TRel Target} \text{ and } A6\): \( \text{enc-reflects-divergence} \)
\end{itemize}
\end{proof}

\begin{defn}
\textbf{def}\( \text{rel: Rel\(=\)indRelRTPO TRel} \)
\end{defn}

\begin{hence}
\( B1\): \( \forall S. (\text{SourceTerm S, TargetTerm ([S])}) \in \text{Rel} \)
\end{hence}

by \( (\text{simp add: indRelRTPO.encR}) \)

\begin{from}
\textbf{from} \( \text{rel have B2: \( \forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (TargetTerm T1, TargetTerm T2) \in \text{Rel} \)} \)
\end{from}

by \( (\text{simp add: indRelRTPO.target}) \)

\begin{from}
\textbf{from} \( \text{rel have B3: \( \forall T1 T2. (TargetTerm T1, TargetTerm T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel\( ^+ \))} \)
\end{from}

by \( (\text{simp add: indRelRTPO-to-TRel(\( \omega \))\{where TRel=TRel\}} \)

\begin{from}
\textbf{from} \( \text{rel have B4: \( \forall S T. (SourceTerm S, TargetTerm T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel\( ^* \))} \)
\end{from}

\begin{itemize}
  \item \textbf{using} \( \text{indRelRTPO-to-TRel(\( \omega \))\{where TRel=TRel\}} \)
  \item \textbf{trans-closure-of-TRel-refl-cond\{where TRel=TRel\}}
\end{itemize}

by \( \text{simp} \)

\begin{assume}
\textbf{assume operational-complete (TRel\( ^* \))}
\end{assume}

\begin{itemize}
  \item \textbf{and} \( \text{operational-sound (TRel\( ^* \))} \)
  \item \textbf{and weak-reduction-simulation (TRel\( ^+ \)) Target} \)
  \item \( \forall P Q Q'. (P, Q) \in \text{TRel\( ^* \) \& Q \rightarrow Target* Q'} \)
  \item \( \rightarrow (\exists P'. P \rightarrow Target* P' \& (P', Q') \in \text{TRel\( ^+ \))} \)
\end{itemize}

\begin{with}
\textbf{with} \( \text{rel have B5: \text{weak-reduction-bisimulation Rel (STCal Source Target))} \)
\end{with}

\begin{using}
\textbf{using} \( \text{OC-iff-indRelRTPO-is-weak-reduction-bisimulation\{where TRel=TRel\}} \)
\end{using}

by \( \text{simp} \)

\begin{from}
\textbf{from} \( \text{rel A1 A2 A3 A4 have B6: \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB)} \{success\}} \)
\end{from}

\begin{itemize}
  \item \textbf{using} \( \text{enc-and-TRel-impl-indRelRTPO-weakly-respects-success\{where TRel=TRel\}} \)
  \item \textbf{and} \( \text{success=success} \)
\end{itemize}

by \( \text{blast} \)

\begin{from}
\textbf{from} \( \text{rel A5 A6 have B7: \text{rel-reflects-divergence Rel (STCal Source Target))} \)
\end{from}

\begin{itemize}
  \item \textbf{using} \( \text{enc-and-TRelImpl-indRelRTPO-reflect-divergence\{where TRel=TRel\}} \)
\end{itemize}
by blast

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

\( \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}) \)

\( \land (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+ \)

\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)

\( \land \text{weak-reduction-bisimulation Rel (STCal Source Target)} \)

(\( \land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}} \)

apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast

next

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

\( \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}) \)

\( \land (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+ \)

\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)

\( \land \text{weak-reduction-bisimulation Rel (STCal Source Target)} \)

(\( \land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}} \)

by auto

from this obtain \( \text{Rel where C1: } \forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel} \)

and \( C2: (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}) \)

and \( C3: (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+ \)

and \( C4: (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)

and \( C5: \text{weak-reduction-bisimulation Rel (STCal Source Target)} \)

and \( C6: \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}} \)

and \( C7: \text{rel-reflects-divergence Rel (STCal Source Target)} \)

by auto

from \( C1 C2 C3 C4 C5 \) have \( \exists \text{Rel.} (\forall S. (\text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)

\( \land (\forall T_1 T_2. (T_1, T_2) \in \text{TRel} \rightarrow (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel}) \)

\( \land (\forall T_1 T_2. (\text{TargetTerm } T_1, \text{TargetTerm } T_2) \in \text{Rel} \rightarrow (T_1, T_2) \in \text{TRel}^+ \)

\( \land (\forall S T. (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^+ \)

\( \land \text{weak-reduction-bisimulation Rel (STCal Source Target)} \)

by blast

hence operational-corresponding (TRel^+)

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

using OC-iff-weak-reduction-bisimulation[where TRel=TRel]

by auto

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

\( \land \text{rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}} \)

\( \land \text{rel-reflects-divergence Rel (STCal Source Target)} \)

apply (rule exI) using C1 C6 C7 by blast

hence enc-weakly-respects-barb-set \{success\} \( \land \text{enc-reflects-divergence} \)

using WSS-DR-iff-source-target-rel

by auto

moreover have \( \text{rel-weakly-respects-barb-set TRel TWB \{success\}} \)

proof auto

fix TP TQ TP'

assume \( (TP, TQ) \in \text{TRel} \)

with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)

by simp

moreover assume \( TP \rightarrow (\text{Calculus TWB})* TP' \land TP'\downarrow<\text{TWB}> \text{success} \)

hence \( \text{TargetTerm } TP\downarrow<\text{STCalWB SWB TWB}> \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)

by blast

ultimately have \( \text{TargetTerm } TQ\downarrow<\text{STCalWB SWB TWB}> \text{success} \)

using \( C6 \)

by blast

thus \( TQ\downarrow<\text{TWB}> \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)

by blast

next

fix TP TQ TQ'

assume \( (TP, TQ) \in \text{TRel} \)
with \( C2 \) have \((\text{TargetTerm TP}, \text{TargetTerm TQ}) \in \text{Rel}\)
by simp

moreover assume \( TQ \rightarrow (\text{Calculus TWB})* \text{TQ'}<\text{TWB}>\text{success}\)

hence TargetTerm TQ\(<\text{STCalWB SWB TWB}>\text{success}\)
using STCalWB-reachesBarbST
by blast

ultimately have TargetTerm TP\(<\text{STCalWB SWB TWB}>\text{success}\)
using C6
by blast

thus \( \text{TP}<\text{TWB}>\text{success}\)
using STCalWB-reachesBarbST
by blast

qed

moreover from \( C2\ C7\) have rel-reflects-divergence \(\text{TRel}\ \text{Target}\)
using STCal-divergent(2)
by blast

ultimately show operational-corresponding \((\text{TRel}^*)\)
\(\land\) weak-reduction-bisimulation \((\text{TRel}^*)\) Target
\(\land\) enc-weakly-respects-barb-set \{success\} \(\land\) rel-weakly-respects-barb-set \(\text{TRel}\ TWB\ \{\text{success}\}\)
\(\land\) enc-reflects-divergence \(\land\) rel-reflects-divergence \(\text{TRel}\ \text{Target}\)
by fast

qed

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

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

shows (weakly-operational-corresponding \(\text{TRel}\ \land\ \text{preorder} \text{TRel}\)
\(\land\) weak-reduction-correspondence-simulation \(\text{TRel}\ \text{Target}\)
\(\land\) enc-weakly-respects-barb-set \{success\}
\(\land\) rel-weakly-respects-barb-set \(\text{TRel}\ TWB\ \{\text{success}\}\)
\(\land\) enc-reflects-divergence \(\land\) rel-reflects-divergence \(\text{TRel}\ \text{TRel}\ \text{Target}\)

\(= (\exists\ \text{Rel}. \ (\forall\ S. (\text{SourceTerm} S, \text{TargetTerm} (\{S\})) \in \text{Rel})
\(\land\) \(\text{TRel} = \{(T1, T2), (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}\)
\(\land\) \(\forall\ S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow (\{S\}, T) \in \text{TRel}\)
\(\land\) weak-reduction-correspondence-simulation \(\text{Rel}(\text{STCal Source Target})\ \land\ \text{preorder} \text{Rel}\)
\(\land\) rel-weakly-respects-barb-set \(\text{Rel}(\text{STCalWB SWB TWB})\ \{\text{success}\}\)
\(\land\) rel-reflects-divergence \(\text{Rel}(\text{STCal Source Target})\))

proof (rule iffI, (erule conjE)+)

assume A1: rel-weakly-preserves-barb-set \(\text{TRel}\ TWB\ \{\text{success}\}\)
and A2: rel-weakly-reflects-barb-set \(\text{TRel}\ TWB\ \{\text{success}\}\)
and A3: enc-weakly-preserves-barb-set \{success\}
and A4: enc-weakly-reflects-barb-set \{success\}
and A5: rel-reflects-divergence \(\text{TRel}\ \text{Target}\) and A6: enc-reflects-divergence
and A7: preorder \(\text{TRel}\)

from A7 have A8: \(\text{TRel}^+ = \text{TRel}\)
using trancl-id[of \(\text{TRel}\)]

unfolding preorder-on-def
by blast

from A7 have A9: \(\text{TRel}^* = \text{TRel}\)
using refl-trancl[of \(\text{TRel}\)] trancl-id[of \(\text{TRel}\)]

unfolding preorder-on-def refl-on-def
by auto

def rel: \(\text{Rel}=\text{indRelRTPO} \text{TRel}\)

hence B1: \(\forall\ S. (\text{SourceTerm} S, \text{TargetTerm} (\{S\})) \in \text{Rel}\)
by (simp add: indRelRTPO.encR)

from rel A8 have B2: \(\text{TRel} = \{(T1, T2), (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}\)
using indRelRTPO-to-TRel(4)[where \(\text{TRel}=\text{TRel}\)]
by (auto simp add: indRelRTPO.target)

from rel A9 have B3: \(\forall\ S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow (\{S\}, T) \in \text{TRel}\)
using indRelRTPO-to-TRel(2)[where \(\text{TRel}=\text{TRel}\)]

trans-closure-of-TRel-refl-cond[where \(\text{TRel}=\text{TRel}\)]
by simp

assume operational-complete TRel and weak-operational-sound TRel and preorder TRel
and weak-reduction-simulation TRel Target
and \(\forall P \; Q \; Q', \; (P, \; Q) \in TRel \land Q \longrightarrow Target \; Q'\)
\(\longrightarrow (\exists P'' \; Q'' \; P \longrightarrow Target \; P'' \land (P'', \; Q'') \in TRel)\)

with rel A8 A9 have B4: weak-reduction-correspondence-simulation Rel (STCal Source Target)
using WOC-iff-indRelRTPO-is-reduction-correspondence-simulation[where TRel=TRel]
by simp

from rel A7 have B5: preorder Rel
using indRelRTPO-is-preorder[where TRel=TRel]

unfolding preorder-on-def
by simp

from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel]
and success\(\Rightarrow success\)
by blast

from rel A5 A6 have B7: rel-reflects-divergence Rel (STCal Source Target)
using enc-and-TRel-impl-indRelRTPO-reflect-divergence[where TRel=TRel]
by blast

show \(\exists Rel. \; (\forall S. \; (SourceTerm \; S, \; TargetTerm \; ([S])) \in Rel)\)
\(\land TRel = \{(T1, \; T2), \; (TargetTerm \; T1, \; TargetTerm \; T2) \in Rel\}\)
\(\land (\forall S \; T. \; (SourceTerm \; S, \; TargetTerm \; T) \in Rel \longrightarrow ([S], \; T) \in TRel)\)
\(\land weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel\)
\(\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}\)
\(\land rel-reflects-divergence Rel (STCal Source Target)\)

apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast

next
assume \(\exists Rel. \; (\forall S. \; (SourceTerm \; S, \; TargetTerm \; ([S])) \in Rel)\)
\(\land TRel = \{(T1, \; T2), \; (TargetTerm \; T1, \; TargetTerm \; T2) \in Rel\}\)
\(\land (\forall S \; T. \; (SourceTerm \; S, \; TargetTerm \; T) \in Rel \longrightarrow ([S], \; T) \in TRel)\)
\(\land weak-reduction-correspondence-simulation Rel (STCal Source Target) \land preorder Rel\)
\(\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}\)
\(\land rel-reflects-divergence Rel (STCal Source Target)\)

from this obtain Rel where C1: \(\forall S. \; (SourceTerm \; S, \; TargetTerm \; ([S])) \in Rel\)
and C2: TRel = \{(T1, \; T2), \; (TargetTerm \; T1, \; TargetTerm \; T2) \in Rel\}
and C3: \(\forall S \; T. \; (SourceTerm \; S, \; TargetTerm \; T) \in Rel \longrightarrow ([S], \; T) \in TRel\)
and C4: weak-reduction-correspondence-simulation Rel (STCal Source Target)
and C5: preorder Rel and C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
and C7: rel-reflects-divergence Rel (STCal Source Target)
by auto

from C1 C2 C3 C4 C5 have \(\exists Rel. \; (\forall S. \; (SourceTerm \; S, \; TargetTerm \; ([S])) \in Rel)\)
\(\land TRel = \{(T1, \; T2), \; (TargetTerm \; T1, \; TargetTerm \; T2) \in Rel\}\)
\(\land (\forall S \; T. \; (SourceTerm \; S, \; TargetTerm \; T) \in Rel \longrightarrow ([S], \; T) \in TRel) \land preorder Rel\)
\(\land weak-reduction-correspondence-simulation Rel (STCal Source Target)\)
by blast

hence weakly-operational-corresponding TRel \land preorder TRel

using WOC-wrt-preorder-iff-reduction-correspondence-simulation[where TRel=TRel]
by simp

moreover have \(\exists Rel. \; (\forall S. \; (SourceTerm \; S, \; TargetTerm \; ([S])) \in Rel)\)
\(\land rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) \{success\}\)
\(\land rel-reflects-divergence Rel (STCal Source Target)\)
apply (rule exI) using C1 C6 C7 by blast

hence enc-weakly-respects-barb-set \{success\} \land enc-reflects-divergence
using WSS-DR-iff-source-target-rel
by simp

moreover have rel-weakly-respects-barb-set TRel TWB \{success\}
proof auto
fix TP TQ TP'
assume (TP, TQ) \in TRel
with C2 have (TargetTerm TP, TargetTerm TQ) \in Rel
by simp
moreover assume \( TP \mapsto \langle (\text{Calculus TWB})^* TP' \rangle <\text{TWB}>\) success
hence TargetTerm \( TP'\downarrow <\text{STCalWB SWB TWB}>\) success
  using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TP'\downarrow <\text{STCalWB SWB TWB}>\) success
  using \( C6 \)
by blast
thus \( TP'\downarrow <\text{TWB}>\) success
  using \( \text{STCalWB-reachesBarbST} \)
by blast

next
fix \( TP TQ TQ' \)
assume \((TP, TQ) \in TRel \)
with \( C2 \) have \((\text{TargetTerm TP}, \text{TargetTerm TQ}) \in \text{Rel} \)
by simp
moreover assume \( TQ \mapsto \langle (\text{Calculus TWB})^* TQ' \rangle <\text{TWB}>\) success
hence TargetTerm \( TQ'\downarrow <\text{STCalWB SWB TWB}>\) success
  using \( \text{STCalWB-reachesBarbST} \)
by blast
ultimately have TargetTerm \( TQ'\downarrow <\text{STCalWB SWB TWB}>\) success
  using \( C6 \)
by blast
thus \( TP\downarrow <\text{TWB}>\) success
  using \( \text{STCalWB-reachesBarbST} \)
by blast
qed

moreover from \( C2 C7 \) have \( \text{rel-reflects-divergence TRel Target} \)
  using \( \text{STCal-divergent(2)} \)
by blast
ultimately show weakly-operational-corresponding TRel \(\land\) preorder TRel
  \(\land\) weak-reduction-correspondence-simulation TRel Target
  \(\land\) enc-weakly-respects-barb-set \{success\} \(\land\) rel-weakly-respects-barb-set TRel TWB \{success\}
  \(\land\) enc-reflects-divergence \(\land\) rel-reflects-divergence TRel Target
by fast
qed

lemma (in encoding-wrt-barbs) OC-SS-DR-wrt-preorder-iff-source-target-rel:
  fixes success :: 'bars
  and TRel :: ('procT \times 'procT) set
  shows (operational-corresponding TRel \(\land\) preorder TRel \(\land\) weak-reduction-bisimulation TRel Target
  \(\land\) enc-weakly-respects-barb-set \{success\}
  \(\land\) rel-weakly-respects-barb-set TRel TWB \{success\}
  \(\land\) enc-reflects-divergence \(\land\) rel-reflects-divergence TRel Target)
  \(\Rightarrow\) \((\exists\text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \text{Rel} \ldots \))
proof (rule iffI, (erule conjE)+)
assume A1: \(\text{rel-weakly-preserves-barb-set TRel TWB \{success\}}\)
  and A2: \(\text{rel-weakly-reflects-barb-set TRel TWB \{success\}}\)
  and A3: \(\text{enc-weakly-preserves-barb-set \{success\}}\)
  and A4: \(\text{enc-weakly-reflects-barb-set \{success\}}\)
  and A5: \(\text{rel-reflects-divergence TRel Target} \text{ and A6: enc-reflects-divergence}\)
  and A7: \(\text{preorder TRel} \)
from A7 have A8: \(\text{TRel}^+ = \text{TRel} \)
  using trancl-id[of TRel]
  unfolding preorder-on-def
by blast
from A7 have A9: TRel* = TRel
  using refl-trancl[of TRel] trancl-id[of TRel]
  unfolding preorder-on-def refl-on-def
by auto
def rel: Rel ≡ indRelRTPO TRel
hence B1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
  by (simp add: indRelRTPO.encR)
from rel A8 have B2: TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2)} ∈ Rel
  using indRelRTPO-to-TRel(4)[where TRel=TRel]
by (auto simp add: indRelRTPO_target)
from rel A9 have B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
  using indRelRTPO-to-TRel(2)[where TRel=TRel]
trans-closure-of-TRel-refl-cond[where TRel=TRel]
by simp
assume operational-complete TRel and operational-sound TRel and preorder TRel
and weak-reduction-simulation TRel Target
and ∀ P Q Q'. (P, Q) ∈ TRel ∧ Q ≡ Target* Q' → (∃ P'. P → Target* P' ∧ (P', Q') ∈ TRel
with rel A8 A9 have B4: weak-reduction-bisimulation Rel (STCal Source Target)
  using OC-iff-indRelRTPO-is-weak-reduction-bisimulation[where TRel=TRel]
by simp
from rel A7 have B5: preorder Rel
  using indRelRTPO-is-preorder[where TRel=TRel]
unfolding preorder-on-def
by simp
from rel A1 A2 A3 A4 have B6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  using enc-and-TRel-impl-indRelRTPO-weakly-respects-success[where TRel=TRel
and success=success]
by blast
from rel A5 A6 have B7: rel-reflects-divergence Rel (STCal Source Target)
  using enc-and-TRelimpl-indRelRTPO-reflect-divergence[where TRel=TRel]
by blast
show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2)} ∈ Rel
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
  ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  ∧ rel-reflects-divergence Rel (STCal Source Target)
apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2)} ∈ Rel
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ weak-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
  ∧ rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
  ∧ rel-reflects-divergence Rel (STCal Source Target)
from this obtain Rel where C1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and C2: TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2)} ∈ Rel
and C3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
and C4: weak-reduction-bisimulation Rel (STCal Source Target) and C5: preorder Rel
and C6: rel-weakly-respects-barb-set Rel (STCalWB SWB TWB) {success}
and C7: rel-reflects-divergence Rel (STCal Source Target)
by auto
from C1 C2 C3 C4 C5 have ∃ Rel.(∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = {(T1, T2), (TargetTerm T1, TargetTerm T2)} ∈ Rel
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel) ∧ preorder Rel
  ∧ weak-reduction-bisimulation Rel (STCal Source Target)
  by blast
hence operational-corresponding TRel ∧ preorder TRel ∧ weak-reduction-bisimulation TRel Target
  using OC-wrt-preorder-iff-weak-reduction-bisimulation[where TRel=TRel]
  by simp
moreover have \( \exists \text{Rel.} (\forall S. \text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel}) \)
\( \land \text{rel-weakly-respects-barb-set} \text{Rel } (\text{STCalWB SWB TWB}) \{\text{success}\} \)
\( \land \text{rel-reflects-divergence} \text{Rel } (\text{STCal Source Target}) \)

apply (rule exI) using \( C1 \ C6 \ C7 \) by blast

hence \( \text{enc-weakly-respects-barb-set} \{\text{success}\} \land \text{enc-reflects-divergence} \)

by simp

moreover have \( \text{rel-weakly-respects-barb-set} \text{TRel TWB} \{\text{success}\} \)

proof auto

fix \( TP \ TQ TP' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by simp

moreover assume \( TP \mapsto (\text{Calculus TWB})\ast \; TP' \land TQ' < \text{TWB} > \text{success} \)

hence \( \text{TargetTerm } TP\downarrow < \text{STCalWB SWB TWB} > \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)
by blast

ultimately have \( \text{TargetTerm } TQ\downarrow < \text{STCalWB SWB TWB} > \text{success} \)

using \( C6 \)
by blast

thus \( TP\downarrow < \text{TWB} > \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)
by blast

next

fix \( TP \ TQ TQ' \)
assume \( (TP, TQ) \in \text{TRel} \)
with \( C2 \) have \( (\text{TargetTerm } TP, \text{TargetTerm } TQ) \in \text{Rel} \)
by simp

moreover assume \( TQ \mapsto (\text{Calculus TWB})\ast \; TQ' \land TQ' < \text{TWB} > \text{success} \)

hence \( \text{TargetTerm } TQ\downarrow < \text{STCalWB SWB TWB} > \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)
by blast

ultimately have \( \text{TargetTerm } TP\downarrow < \text{STCalWB SWB TWB} > \text{success} \)

using \( C6 \)
by blast

thus \( TQ\downarrow < \text{TWB} > \text{success} \)

using \( \text{STCalWB-reachesBarbST} \)
by blast

qed

moreover from \( C2 \ C7 \) have \( \text{rel-reflects-divergence} \text{TRel Target} \)

using \( \text{STCal-divergent}(2) \)
by blast

ultimately

show operational-corresponding \( \text{TRel} \land \text{preorder} \text{TRel} \land \text{weak-reduction-bisimulation} \text{TRel Target} \)
\( \land \text{enc-weakly-respects-barb-set} \{\text{success}\} \land \text{rel-weakly-respects-barb-set} \text{TRel TWB} \{\text{success}\} \)
\( \land \text{enc-reflects-divergence} \land \text{rel-reflects-divergence} \text{TRel Target} \)

by fast

qed

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

fixes \( \text{success} :: \text{'barbs} \)
and \( \text{TRel} :: \text{('procT x 'procT) set} \)
shows (strongly-operational-corresponding \( \text{TRel} \land \text{preorder} \text{TRel} \)
\( \land \text{strong-reduction-bisimulation} \text{TRel Target} \)
\( \land \text{enc-respects-barb-set} \{\text{success}\} \land \text{rel-respects-barb-set} \text{TRel TWB} \{\text{success}\} \)
\( \land \text{enc-reflects-divergence} \land \text{rel-reflects-divergence} \text{TRel Target} \)

= (\exists \text{Rel.} (\forall S. \text{SourceTerm } S, \text{TargetTerm } ([S])) \in \text{Rel})
\( \land \text{TRel} = \{(T1, T2). \; (\text{TargetTerm } T1, \text{TargetTerm } T2) \in \text{Rel}\} \)
\( \land (\forall S \ T. \; (\text{SourceTerm } S, \text{TargetTerm } T) \in \text{Rel} \mapsto ([S], T) \in \text{TRel}) \)
\( \land \text{strong-reduction-bisimulation} \text{Rel } (\text{STCal Source Target}) \land \text{preorder Rel} \)
\( \land \text{rel-respects-barb-set} \text{Rel } (\text{STCalWB SWB TWB}) \{\text{success}\} \)
∧ rel-reflects-divergence Rel (STCal Source Target))

proof (rule iffI, (erule conjE)+)
assume A1: rel-preserves-barb-set TRel TWB \{success\}
and A2: rel-reflects-barb-set TRel TWB \{success\}
and A3: enc-preserves-barb-set \{success\} and A4: enc-reflects-barb-set \{success\}
and A5: rel-reflects-divergence TRel Target and A6: enc-reflects-divergence
and A7: preorder TRel
from A7 have A8: TRel⁺ = TRel
  using trancl-id[of TRel]
  unfolding preorder-on-def
  by blast
from A7 have A9: TRel⁺ = TRel
  using refl-trancl[of TRel] trancl-id[of TRel]
  unfolding preorder-on-def refl-on-def
  by auto
def rel: Rel≡indRelRTPO TRel
hence B1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
by (simp add: indRelRTPO.encR)
from rel A8 have B2: TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel\}
  using indRelRTPO-to-TRel(4)[where TRel=TRel]
by (auto simp add: indRelRTPO.target)
from rel A9 have B3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
  using indRelRTPO-to-TRel(2)[where TRel=TRel]
  trans-closure-of-TRel-refl-cond[where TRel=TRel]
  by simp
assume strongly-operational-complete TRel and strongly-operational-sound TRel
and preorder TRel and strong-reduction-simulation TRel Target
and ∀ P Q Q'. (P, Q) ∈ TRel ∧ Q → Target Q' → (∃ P'. P → Target P' ∧ (P', Q') ∈ TRel)
with rel A8 A9 have B4: strong-reduction-bisimulation Rel (STCal Source Target)
  using SOC-iff-indRelRTPO-is-strong-reduction-bisimulation[where TRel=TRel]
  by simp
from rel A7 have B5: preorder Rel
  using indRelRTPO-is-preorder[where TRel=TRel]
  unfolding preorder-on-def
  by simp
from rel A1 A2 A3 A4 have B6: rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
  using enc-and-Trel-impl-indRelRTPO-respects-success[where TRel=TRel and success=succ]
  by blast
from rel A5 A6 have B7: rel-reflects-divergence Rel (STCal Source Target)
  using enc-and-Trel-impl-indRelRTPO-reflect-divergence[where TRel=TRel]
  by blast
show ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel\}
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
  ∧ rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
  ∧ rel-reflects-divergence Rel (STCal Source Target)
  apply (rule exI) using B1 B2 B3 B4 B5 B6 B7 by blast
next
assume ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
  ∧ TRel = \{(T1, T2). (TargetTerm T1, TargetTerm T2) ∈ Rel\}
  ∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel)
  ∧ strong-reduction-bisimulation Rel (STCal Source Target) ∧ preorder Rel
  ∧ rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
  ∧ rel-reflects-divergence Rel (STCal Source Target)
from this obtain Rel where C1: ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
and C2: TRel = \{(T1, T2), (TargetTerm T1, TargetTerm T2) ∈ Rel\}
and C3: ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel → ([S], T) ∈ TRel
and C4: strong-reduction-bisimulation Rel (STCal Source Target) and C5: preorder Rel
and C6: rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}
and C7: rel-reflects-divergence Rel (STCal Source Target)
by auto
from C1 C2 C3 C4 C5 have \( \exists \text{Rel.} (\forall S. \langle \text{SourceTerm S, TargetTerm ([S])} \rangle \in \text{Rel}) \)
\( \land \text{TRel} = \{(T1, T2). \langle \text{TargetTerm T1, TargetTerm T2} \rangle \in \text{Rel}\} \)
\( \land (\forall S T. \langle \text{SourceTerm S, TargetTerm T} \rangle \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}) \land \text{preorder Rel} \)
\( \land \text{strong-reduction-bisimulation Rel (STCal Source Target)} \)
by blast

hence strongly-operational-corresponding TRel \( \land \text{preorder TRel} \)
\( \land \text{strong-reduction-bisimulation TRel Target} \)
using SOC-wrt-preorder-iff-strong-reduction-bisimulation[where TRel=TRel]
by simp

moreover have \( \exists \text{Rel.} (\forall S. \langle \text{SourceTerm S, TargetTerm ([S])} \rangle \in \text{Rel}) \)
\( \land \text{rel-respects-barb-set Rel (STCalWB SWB TWB) \{success\}} \)
\( \land \text{rel-reflects-divergence Rel (STCal Source Target)} \)
apply (rule exI) using C1 C6 C7 by blast

hence enc-respects-barb-set \{success\} \( \land \text{enc-reflects-divergence} \)
using SS-DR-iff-source-target-rel
by simp

moreover have rel-respects-barb-set TRel TWB \{success\}

proof auto
fix TP TQ
assume \( (TP, TQ) \in \text{TRel} \)
with C2 have \( \langle \text{TargetTerm TP, TargetTerm TQ} \rangle \in \text{Rel} \)
by simp

moreover assume TP\(\downarrow<\text{TWB}\rangle\text{success} \)
hence TargetTerm TP\(\downarrow<\text{STCalWB SWB TWB}\rangle\text{success} \)
using STCalWB-hasBarbST
by blast

ultimately have TargetTerm TQ\(\downarrow<\text{STCalWB SWB TWB}\rangle\text{success} \)
using C6
by blast

thus TQ\(\downarrow<\text{TWB}\rangle\text{success} \)
using STCalWB-hasBarbST
by blast

next
fix TP TQ
assume \( (TP, TQ) \in \text{TRel} \)
with C2 have \( \langle \text{TargetTerm TP, TargetTerm TQ} \rangle \in \text{Rel} \)
by simp

moreover assume TQ\(\downarrow<\text{TWB}\rangle\text{success} \)
hence TargetTerm TQ\(\downarrow<\text{STCalWB SWB TWB}\rangle\text{success} \)
using STCalWB-hasBarbST
by blast

ultimately have TargetTerm TP\(\downarrow<\text{STCalWB SWB TWB}\rangle\text{success} \)
using C6
by blast

thus TP\(\downarrow<\text{TWB}\rangle\text{success} \)
using STCalWB-hasBarbST
by blast

qed

moreover from C2 C7 have rel-reflects-divergence TRel Target
using STCal-divergent(2)
by blast

ultimately show strongly-operational-corresponding TRel \( \land \text{preorder TRel} \)
\( \land \text{strong-reduction-bisimulation TRel Target} \)
\( \land \text{enc-respects-barb-set \{success\}} \land \text{rel-respects-barb-set TRel TWB \{success\}} \)
\( \land \text{enc-reflects-divergence} \land \text{rel-reflects-divergence TRel Target} \)
by fast

qed

271
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** $SRel :: (\text{\textquoteleft}procS \times \text{\textquoteleft}procS\text{\textquoteleft})$ set
- and $TRel :: (\text{\textquoteleft}procT \times \text{\textquoteleft}procT\text{\textquoteleft})$ set
- **Assumes** $eqS$: equivalence $SRel$
- and $eqT$: equivalence $TRel$
- **Shows** fully-abstract $SRel$ $TRel$
  \begin{align*}
  & \land \text{operational-corresponding} \ TRel \land \text{weak-reduction-bisimulation} \ TRel \ Target \\
  \iffreq & (\exists \ Rel).
  \\
  & (\forall \ S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \land \text{(TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel})
  \\
  & \land SRel = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}
  \\
  & \land TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}
  \\
  & \land \text{trans Rel} \land \text{weak-reduction-bisimulation} \text{Rel (STCal Source Target)}
\end{align*}

**Proof** (rule iffI, erule conjE, erule conjE)
- **Assume** $A1$: fully-abstract $SRel$ $TRel$ and $A2$: operational-corresponding $TRel$
- and $A3$: weak-reduction-bisimulation $TRel$ $Target$
- **From** $eqT$ **Have** $A4$: $TRel^* = TRel$
  - using refl-trancl[of $TRel$] trancl-id[of $TRel$]
- **Unfolding** equiv-def refl-on-def
- by auto
- **Have** $A5$:
  \begin{align*}
  & \forall \ S. \text{SourceTerm} \ S \sim [\cdot] T < TRel > \text{TargetTerm} \ ([S]) \land \text{TargetTerm} \ ([S]) \sim [\cdot] T < TRel > \text{SourceTerm} \ S \\
  & \text{by (simp add: indRelTEQ.encR indRelTEQ.encL)}
\end{align*}
- **Moreover from** $A4$ **Have** $A6$: $TRel = \{(T1, T2). \text{TargetTerm} \ T1 \sim [\cdot] T < TRel > \text{TargetTerm} \ T2\}$
  - using indRelTEQ-to-TRel(4)[where $TRel=TRel$]
  - trans-closure-of-TRel-refl-cond[where $TRel=TRel$]
- by (auto simp add: indRelTEQ.target)
- **Moreover have** $A7$: $\exists \ S. \text{SourceTerm} \ S \sim [\cdot] T < TRel > \text{TargetTerm} \ ([S])
  \land \text{TargetTerm} \ ([S]) \sim [\cdot] T < TRel > \text{SourceTerm} \ S$
- **Unfolding** trans-def
- by blast
- **Moreover have** $SRel = \{(S1, S2). \text{SourceTerm} \ S1 \sim [\cdot] T < TRel > \text{SourceTerm} \ S2\}$
- **Proof** –
  - from $A6$ **Have** $\forall \ S. \text{SourceTerm} \ S \sim [\cdot] T < TRel > \text{SourceTerm} \ S2$
  - by blast
- **Moreover have** $\text{indRelTEQ TRel} \cup \{(P, Q). \exists S. [S] \in T P \land S \in S Q\} = \text{indRelTEQ TRel}$
- by (auto simp add: indRelTEQ.encL)
- **With** $A7$ **Have** $\exists \ S. \text{SourceTerm} \ S \sim [\cdot] T < TRel > \text{SourceTerm} \ S2$
- **Unfolding** trans-def
- by blast
- **Ultimately show** $SRel = \{(S1, S2). \text{SourceTerm} \ S1 \sim [\cdot] T < TRel > \text{SourceTerm} \ S2\}$
  - using $A1$ $A5$ full-abstraction-and-trans-relation-contains-TRel-impl-SRel[where $SRel=SRel$ and $TRel=TRel$ and $Rel=indRelTEQ TRel$]
  - by blast
- qed

**Moreover from** $eqT$ **A2 A3 have** weak-reduction-bisimulation $(\text{indRelTEQ TRel})$ $(\text{STCal Source Target})$
  - using OC-wrt-equivalence-iff-indRelTEQ-weak-reduction-bisimulation[where $TRel=TRel$]
  - by blast
- **Ultimately show** $\exists \ Rel. \ (\forall \ S. (\text{SourceTerm} \ S, \text{TargetTerm} \ ([S])) \in \text{Rel} \land (\text{TargetTerm} \ ([S]), \text{SourceTerm} \ S) \in \text{Rel})$
  - $\land SRel = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\}$
  - $\land TRel = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\}$
  - $\land \text{trans Rel} \land \text{weak-reduction-bisimulation} \text{Rel (STCal Source Target)}$
  - by blast

next
assume
\( \exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \land (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel}) \)
\( \wedge \text{SRel} = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\} \)
\( \wedge \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\} \)
\( \wedge \text{trans Rel} \land \text{weak-reduction-bisimulation Rel} (\text{STCal Source Target}) \)

from this obtain Rel where
  B1: \( \forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel} \land (\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel} \)
  and B2: \( \text{SRel} = \{(S1, S2). (\text{SourceTerm} S1, \text{SourceTerm} S2) \in \text{Rel}\} \)
  and B3: \( \text{TRel} = \{(T1, T2). (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\} \)
  and B4: trans Rel and B5: weak-reduction-bisimulation Rel (STCal Source Target)

by blast

moreover have operational-corresponding TRel \(\land\) weak-reduction-bisimulation TRel Target

proof –
from eqT have C1: \(\text{TRel}^+ = \text{TRel}\)
  using trancl-id[of TRel]
  unfolding equiv-def
  by blast

from eqT have C2: \(\text{TRel}^* = \text{TRel}\)
  using reflcl-trancl[of TRel] trancl-id[of TRel]
  unfolding equiv-def refl-on-def
  by auto

from B1 have \(\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}\)
  by simp

moreover from B3 have \(\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}\)
  by simp

moreover from B3 C1 have \(\forall T1 T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+\)
  by simp

moreover have \(\forall S T. (\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel} \rightarrow ([S], T) \in \text{TRel}^*\)

proof clarify
fix S T
from B1 have \(\text{TargetTerm} ([S]), \text{SourceTerm} S) \in \text{Rel}\)
  by simp

moreover assume \(\text{SourceTerm} S, \text{TargetTerm} T) \in \text{Rel}\)
ultimately have \(\text{TargetTerm} ([S]), \text{TargetTerm} T) \in \text{Rel}\)
  using B4
  unfolding trans-def
  by blast
with B3 C2 show \(([S], T) \in \text{TRel}^*\)
  by simp

qed

ultimately have \(\exists \text{Rel}. (\forall S. (\text{SourceTerm} S, \text{TargetTerm} ([S])) \in \text{Rel}) \)
\(\wedge (\forall T1 T2. (T1, T2) \in \text{TRel} \rightarrow (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel}) \)
\(\wedge (\forall T1 T2. (\text{TargetTerm} T1, \text{TargetTerm} T2) \in \text{Rel} \rightarrow (T1, T2) \in \text{TRel}^+) \)
\(\wedge \text{weak-reduction-bisimulation Rel} (\text{STCal Source Target}) \)
  using B5
  by blast

with C1 C2 show operational-corresponding TRel \(\land\) weak-reduction-bisimulation TRel Target
  using OC-iff-weak-reduction-bisimulation[where TRel=TRel]
  by auto

qed

ultimately show fully-abstract SRel TRel \(\land\) operational-corresponding TRel
  \(\land\) weak-reduction-bisimulation TRel Target
  by simp

lemma (in encoding) FA-SOC-modulo-equivalences-iff-source-target-relation:
fixes $S_{Rel}$ :: ("proc$k S" × "proc$k S") set
and $T_{Rel}$ :: ("proc$k T" × "proc$k T") set
assumes $eq_S$: equivalence $S_{Rel}$
and $eq_T$: equivalence $T_{Rel}$
sows fully-abstract $S_{Rel}$ $T_{Rel}$ ∧ strongly-operational-corrresponding $T_{Rel}$
∧ strong-reduction-bisimulation $T_{Rel}$ Target $\leftrightarrow$ (∃ $Rel$.
(∀ $S$. ($SourceTerm$ $S$, $TargetTerm$ [: $S$]) ∈ $Rel$ ∧ ($TargetTerm$ [: $S$], $SourceTerm$ $S$) ∈ $Rel$)
∧ $S_{Rel}$ = {(S1, S2). ($SourceTerm$ S1, $SourceTerm$ S2) ∈ $Rel$}
∧ $T_{Rel}$ = {(T1, T2). ($TargetTerm$ T1, $TargetTerm$ T2) ∈ $Rel$} ∧ trans $Rel$
∧ strong-reduction-bisimulation $Rel$ (STCal Source Target))

proof (rule iffI, erule conjE, erule conjE)
assume A1: fully-abstract $S_{Rel}$ $T_{Rel}$ and A2: strongly-operational-corrresponding $T_{Rel}$
and A3: strong-reduction-bisimulation $T_{Rel}$ Target
from $eq_T$ have A4: $T_{Rel}$" $T_{Rel}$
using refl-trancl[of $T_{Rel}$] trancl-id[of $T_{Rel}$]
unfolding equiv-def refl-on-def
by auto
have A5:
∀ $S$. ($SourceTerm$ $S$ $\sim$ [: $T$ < $T_{Rel}$ >] $TargetTerm$ [: $S$]) ∧ $TargetTerm$ [: $S$] $\sim$ [: $T$ < $T_{Rel}$ >] $SourceTerm$ $S$
by (simp add: indRelTEQ_encR indRelTEQ_encL)
moreover have A6: $T_{Rel}$ = {(T1, T2). $TargetTerm$ T1 $\sim$ [: $T$ < $T_{Rel}$ >] $TargetTerm$ T2}
using indRelTEQ-to-$T_{Rel}$(4)[where $T_{Rel}$=$T_{Rel}$]
trans-closure-of-$T_{Rel}$-refl-cond[where $T_{Rel}$=$T_{Rel}$]
by (auto simp add: indRelTEQ_target)
moreover have A7: trans (indRelTEQ $T_{Rel}$)
using indRelTEQ.trans[where $T_{Rel}$=$T_{Rel}$]
unfolding trans-def
by blast
moreover have $S_{Rel}$ = {(S1, S2). $SourceTerm$ S1 $\sim$ [: $T$ < $T_{Rel}$ >] $SourceTerm$ S2}
proof -
from A6 have ∀ S1 S2. ([(S1), [S2]]) ∈ $T_{Rel}$) = $TargetTerm$ [(S1)] $\sim$ [: $T$ < $T_{Rel}$ >] $TargetTerm$ [(S2)]
by blast
moreover have indRelTEQ $T_{Rel}$∪ {(P, Q). ⟨S. [S] ∈ T P ∧ S ∈ S Q} = indRelTEQ $T_{Rel}$
by (auto simp add: indRelTEQ_encL)
with A7 have trans (indRelTEQ $T_{Rel}$∪ {(P, Q). ⟨S. [S] ∈ T P ∧ S ∈ S Q})
unfolding trans-def
by blast
ultimately show $S_{Rel}$ = {(S1, S2). $SourceTerm$ S1 $\sim$ [: $T$ < $T_{Rel}$ >] $SourceTerm$ S2}
using A1 A5 full-abstraction-and-trans-relation-contains-$T_{Rel}$-impl-$S_{Rel}$(where
$S_{Rel}$=$S_{Rel}$ and $T_{Rel}$=$T_{Rel}$ and $Rel$=indRelTEQ $T_{Rel}$)
by blast
qed
moreover from $eq_T$ A2 A3 have strong-reduction-bisimulation (indRelTEQ $T_{Rel}$) (STCal Source Target)
using SOC-wrt-equivalence-iff-indRelTEQ-strong-reduction-bisimulation[where $T_{Rel}$=$T_{Rel}$]
by blast
ultimately
show ⟨∃ $Rel$. (⟨∀ S. ($SourceTerm$ $S$, $TargetTerm$ [: $S$]) ∈ $Rel$ ∧ ($TargetTerm$ [: $S$], $SourceTerm$ $S$) ∈ $Rel$)
∧ $S_{Rel}$ = {(S1, S2). ($SourceTerm$ S1, $SourceTerm$ S2) ∈ $Rel$}
∧ $T_{Rel}$ = {(T1, T2). ($TargetTerm$ T1, $TargetTerm$ T2) ∈ $Rel$} ∧ trans $Rel$
∧ strong-reduction-bisimulation $Rel$ (STCal Source Target)
by blast
next
assume ⟨∃ $Rel$. (⟨∀ S. ($SourceTerm$ $S$, $TargetTerm$ [: $S$]) ∈ $Rel$ ∧ ($TargetTerm$ [: $S$], $SourceTerm$ $S$) ∈ $Rel$)
∧ $S_{Rel}$ = {(S1, S2). ($SourceTerm$ S1, $SourceTerm$ S2) ∈ $Rel$}
∧ $T_{Rel}$ = {(T1, T2). ($TargetTerm$ T1, $TargetTerm$ T2) ∈ $Rel$} ∧ trans $Rel$
∧ strong-reduction-bisimulation $Rel$ (STCal Source Target)
from this obtain $Rel$ where
B1: ⟨∀ S. ($SourceTerm$ $S$, $TargetTerm$ [: $S$]) ∈ $Rel$ ∧ ($TargetTerm$ [: $S$], $SourceTerm$ $S$) ∈ $Rel$
and B2: $S_{Rel}$ = {(S1, S2). ($SourceTerm$ S1, $SourceTerm$ S2) ∈ $Rel$}
and B3: $T_{Rel}$ = {(T1, T2). ($TargetTerm$ T1, $TargetTerm$ T2) ∈ $Rel}$ and B4: trans $Rel$
and B5: strong-reduction-bisimulation Rel (STCal Source Target)
by blast
from B1 B2 B3 B4 have fully-abstract SRel TRel
using trans-source-target-relation-impl-fully-abstract[where Rel=Rel and SRel=SRel
and TRel=TRel]
by blast
moreover
have strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target
proof –
from eqT have C1: TRel⁺ = TRel
using trancl-id[of TRel]
unfolding equiv-def refl-on-def
by blast
from eqT have C2: TRel⁺⁺ = TRel
using reflcl-trancl[of TRel] trancl-id[of TRel]
unfolding equiv-def refl-on-def
by auto
from B1 have ∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel
by simp
moreover from B3 have ∀ T1 T2. (T1, T2) ∈ TRel →→ (TargetTerm T1, TargetTerm T2) ∈ Rel
by simp
moreover from B3 C1
have ∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel →→ (T1, T2) ∈ TRel⁺⁺
by simp
moreover have ∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel →→ ([S], T) ∈ TRel⁺⁺
proof clarify
fix S T
from B1 have (TargetTerm ([S]), SourceTerm S) ∈ Rel
by simp
moreover assume (SourceTerm S, TargetTerm T) ∈ Rel
ultimately have (TargetTerm ([S]), TargetTerm T) ∈ Rel
using B4
unfolding trans-def
by blast
with B3 C2 show ([S], T) ∈ TRel⁺⁺
by simp
qed
ultimately have ∃ Rel. (∀ S. (SourceTerm S, TargetTerm ([S])) ∈ Rel)
∧ (∀ T1 T2. (T1, T2) ∈ TRel →→ (TargetTerm T1, TargetTerm T2) ∈ Rel)
∧ (∀ T1 T2. (TargetTerm T1, TargetTerm T2) ∈ Rel →→ (T1, T2) ∈ TRel⁺⁺)
∧ (∀ S T. (SourceTerm S, TargetTerm T) ∈ Rel →→ ([S], T) ∈ TRel⁺⁺)
∧ strong-reduction-bisimulation Rel (STCal Source Target)
using B5
by blast
with C1 C2
show strongly-operational-corresponding TRel ∧ strong-reduction-bisimulation TRel Target
using SOC-iff-strong-reduction-bisimulation[where TRel=TRel]
by auto
qed
ultimately show fully-abstract SRel TRel ∧ strongly-operational-corresponding TRel
∧ strong-reduction-bisimulation TRel Target
by simp
qed
An encoding that is fully abstract w.r.t. the equivalences SRel and TRel and operationally correspond-
ing w.r.t. TRel ensures that SRel is a bisimulation iff TRel is a bisimulation.

lemma (in encoding) FA-and-OC-and-TRel-impl-SRel-bisimulation:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel
and opCom: operational-complete TRel
and opSou: operational-sound TRel
and symmT: sym TRel
and transT: trans TRel
and bisimT: weak-reduction-bisimulation TRel Target
shows weak-reduction-bisimulation SRel Source

proof auto
fix SP SQ SP'
assumes (SP', SQ') ∈ SRel

with opCom obtain TP' where A1: [SP] ⟷ Target TP' and A2: ([SP'], TP') ∈ TRel
by simp

with bisimT A1 obtain TQ' where A3: [SQ] ⟷ Target TQ' and A4: (TP', TQ') ∈ TRel
by blast

from A3 opSou obtain SQ' where A5: SQ ⟷ Source SQ' and A6: ([SQ'], TQ') ∈ TRel
by blast

from A2 A4 A6 symmT transT have ([SP'], [SQ']) ∈ TRel

unfolding trans-def sym-def
by blast

with fullAbs A5 show ∃ SQ': SQ ⟷ Source SQ' ∧ (SP', SQ') ∈ SRel
by blast

next
fix SP SQ SQ'
assumes (SP', SQ') ∈ SRel

with opCom obtain TQ' where B1: [SQ] ⟷ Target TQ' and B2: ([SQ'], TQ') ∈ TRel
by simp

with bisimT B1 obtain TP' where B3: [SP] ⟷ Target TP' and B4: (TP', TQ') ∈ TRel
by blast

from B3 opSou obtain SP' where B5: SP ⟷ Source SP' and B6: ([SP'], TP') ∈ TRel
by blast

from B2 B4 B6 symmT transT have ([SP'], [SQ']) ∈ TRel

unfolding trans-def sym-def
by blast

with fullAbs B5 show ∃ SP': SP ⟷ Source SP' ∧ (SP', SQ') ∈ SRel
by blast

qed

lemma (in encoding) FA-and-SOC-and-TRel-impl-SRel-strong-bisimulation:
fixes SRel :: ('procS × 'procS) set
and TRel :: ('procT × 'procT) set
assumes fullAbs: fully-abstract SRel TRel

and opCom: strongly-operational-complete TRel
and opSou: strongly-operational-sound TRel
and symmT: sym TRel
and transT: trans TRel
and bisimT: weak-reduction-bisimulation TRel Target
shows strong-reduction-bisimulation SRel Source

proof auto
fix SP SQ SP'
assumes (SP', SQ') ∈ SRel

with opCom obtain TP' where A1: [SP] ⟷ Target TP' and A2: ([SP'], TP') ∈ TRel
by blast

with bisimT A1 obtain TQ' where A3: [SQ] ⟷ Target TQ' and A4: (TP', TQ') ∈ TRel
by simp
proof
from A3 opSOU obtain SQ' where A5: SQ \rightarrow Source SQ' and A6: ([SQ'], TQ') \in TRel
by blast

next
fix SP SQ SQ'
assume SQ \rightarrow Source SQ'

from opCom obtain TQ' where B1: [SQ] \rightarrow Target TQ' and B2: ([SQ'], TQ') \in TRel
by blast

with fullAbs have ([SP], [SQ]) \in TRel
by simp

with bisimT B1 obtain TP' where B3: [SP] \rightarrow Target TP' and B4: (TP', TQ') \in TRel
by blast

from B3 opSOU obtain SP' where B5: SP \rightarrow Source SP' and B6: ([SP'], TP') \in TRel
by blast

from B2 B4 B6 symmT transT have ([SP'], [SQ']) \in TRel
unfolding trans-def sym-def
by blast

with fullAbs B5 show \exists SP'. SP \rightarrow Source SP' \land (SP', SQ') \in SRel
by blast

qed

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

fixes SRel :: ('procS \times 'procS) set
and TRel :: ('procT \times 'procT) set
assumes fullAbs: fully-abstract SRel TRel
and opCor: operational-corresponding TRel
and symmT: sym TRel
and transT: trans TRel
and surj: \forall T. \exists S. T = [S]

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

proof
assume bisimS: weak-reduction-bisimulation SRel Source
have weak-reduction-simulation TRel Target

proof clarify
fix TP TQ TP'
from surj have \exists S. TP = [S]
by simp
from this obtain SP where A1: [SP] = TP
by blast
from surj have \exists S. TQ = [S]
by simp
from this obtain SQ where A2: [SQ] = TQ
by blast
assume TP \rightarrow Target* TP'
with opCor A1 obtain SP' where A3: SP \rightarrow Source* SP' and A4: ([SP'], TP') \in TRel
by blast
assume (TP, TQ) \in TRel
with fullAbs A1 A2 have (SP, SQ) \in SRel
by simp
with bisimS A3 obtain SQ' where A5: SQ \rightarrow Source* SQ' and A6: (SP', SQ') \in SRel
by blast
from opCor A2 A5 obtain TQ' where A7: TQ \rightarrow Target* TQ' and A8: ([SQ'], TQ') \in TRel
by blast
from symmT A4 have (TP', [SP']) \in TRel
unfolding sym-def

277
by simp

moreover from fullAbs A6 have $[[SP^'], [SQ']] \in TRel$

by simp

ultimately have $(TP', TQ') \in TRel$

using transT A8

unfolding trans-def

by blast

with A7 show $\exists TQ', TQ \rightarrow\rightarrow Target^* \land (TP', TQ') \in TRel$

by blast

qed

with symmT show weak-reduction-bisimulation $TRel \rightarrow Target$

using symm-weak-reduction-simulation-is-bisimulation[where Rel=TRel and Cal=Target]

by blast

next

assume weak-reduction-bisimulation $TRel \rightarrow Target$

with fullAbs opCor symmT transT show weak-reduction-bisimulation $SRel \rightarrow Source$

using FA-and-OC-and-TRel-impl-SRel-bisimulation[where SRel=SRel and TRel=TRel]

by blast

qed

end