

# Conformance Relations between Input/Output Languages

Robert Sachtleben

March 17, 2025

## Abstract

This entry formalises the paper of the same name by Huang et al. [1] and presents a unifying characterisation of well-known conformance relations such as equivalence and language inclusion (reduction) on languages over input/output pairs. This characterisation simplifies comparisons between conformance relations and from it a fundamental necessary and sufficient criterion for conformance testing is developed.

## Contents

<b>1</b>	<b>Preliminaries</b>	<b>2</b>
<b>2</b>	<b>Conformance Relations</b>	<b>4</b>
<b>3</b>	<b>Unifying Characterisations</b>	<b>5</b>
3.1	$\preceq$ Conformance . . . . .	5
3.2	$\leq$ Conformance . . . . .	7
<b>4</b>	<b>Comparing Conformance Relations</b>	<b>8</b>
4.1	Completely Specified Languages . . . . .	9
<b>5</b>	<b>Conformance Testing</b>	<b>9</b>
<b>6</b>	<b>Reductions Between Relations</b>	<b>11</b>
6.1	Quasi-Equivalence via Quasi-Reduction and Absences . . . . .	11
6.2	Quasi-Reduction via Reduction and explicit Undefined Behaviour . . . . .	12
6.3	Strong Reduction via Reduction and Undefinedness Outputs	13
<b>theory</b>	<i>Input-Output-Language-Conformance</i>	
<b>imports</b>	<i>HOL-Library.Sublist</i>	
<b>begin</b>		

# 1 Preliminaries

**type-synonym** ('a) *alphabet* = 'a set  
**type-synonym** ('x, 'y) *word* = ('x × 'y) list  
**type-synonym** ('x, 'y) *language* = ('x, 'y) word set  
**type-synonym** ('y) *output-relation* = ('y set × 'y set) set

**fun** *is-language* :: 'x alphabet ⇒ 'y alphabet ⇒ ('x, 'y) language ⇒ bool **where**  
  *is-language* X Y L = (  
    — nonempty  
    (L ≠ {}) ∧  
    (∀ π ∈ L .  
      — over X and Y  
      (∀ xy ∈ set π . fst xy ∈ X ∧ snd xy ∈ Y) ∧  
      — prefix closed  
      (∀ π' . prefix π' π → π' ∈ L)))

**lemma** *language-contains-nil* :  
  **assumes** *is-language* X Y L  
**shows** [] ∈ L  
  ⟨proof⟩

**lemma** *language-intersection-is-language* :  
  **assumes** *is-language* X Y L1  
  **and**    *is-language* X Y L2  
**shows** *is-language* X Y (L1 ∩ L2)  
  ⟨proof⟩

**fun** *language-for-state* :: ('x, 'y) language ⇒ ('x, 'y) word ⇒ ('x, 'y) language **where**  
  *language-for-state* L π = {τ . π@τ ∈ L}

**notation** *language-for-state* (⟨ $\mathcal{L}[-, -]$ ⟩)

**lemma** *language-for-state-is-language* :  
  **assumes** *is-language* X Y L  
  **and**    π ∈ L  
**shows** *is-language* X Y  $\mathcal{L}[L, \pi]$   
  ⟨proof⟩

**lemma** *language-of-state-empty-iff* :  
  **assumes** *is-language* X Y L  
**shows** ( $\mathcal{L}[L, \pi] = \{\}$ ) ↔ (π ∉ L)  
  ⟨proof⟩

**fun** *are-equivalent-for-language* :: ('x,'y) language  $\Rightarrow$  ('x,'y) word  $\Rightarrow$  ('x,'y) word  
 $\Rightarrow$  bool **where**  
*are-equivalent-for-language* L  $\alpha$   $\beta$  = ( $\mathcal{L}[L,\alpha] = \mathcal{L}[L,\beta]$ )

**abbreviation**(*input*) *input-projection*  $\pi \equiv \text{map fst } \pi$   
**abbreviation**(*input*) *output-projection*  $\pi \equiv \text{map snd } \pi$   
**notation** *input-projection* ( $\langle[-]_I\rangle$ )  
**notation** *output-projection* ( $\langle[-]_O\rangle$ )

**fun** *is-executable* :: ('x,'y) language  $\Rightarrow$  ('x,'y) word  $\Rightarrow$  'x list  $\Rightarrow$  bool **where**  
*is-executable* L  $\pi$  xs = ( $\exists \tau \in \mathcal{L}[L,\pi] . [\tau]_I = xs$ )

**fun** *executable-sequences* :: ('x,'y) language  $\Rightarrow$  ('x,'y) word  $\Rightarrow$  'x list set **where**  
*executable-sequences* L  $\pi$  = {xs . *is-executable* L  $\pi$  xs}

**fun** *executable-inputs* :: ('x,'y) language  $\Rightarrow$  ('x,'y) word  $\Rightarrow$  'x set **where**  
*executable-inputs* L  $\pi$  = {x . *is-executable* L  $\pi$  [x]}

**notation** *executable-inputs* ( $\langle \text{exec}[-,-] \rangle$ )

**lemma** *executable-sequences-alt-def* : *executable-sequences* L  $\pi$  = {xs .  $\exists$  ys . *length*  
ys = *length* xs  $\wedge$  *zip* xs ys  $\in \mathcal{L}[L,\pi]$ }  
 $\langle \text{proof} \rangle$

**lemma** *executable-inputs-alt-def* : *executable-inputs* L  $\pi$  = {x .  $\exists$  y . [(x,y)]  $\in$   
 $\mathcal{L}[L,\pi]$ }  
 $\langle \text{proof} \rangle$

**lemma** *executable-inputs-in-alphabet* :  
**assumes** *is-language* X Y L  
**and** x  $\in$  *exec*[L, $\pi$ ]  
**shows** x  $\in$  X  
 $\langle \text{proof} \rangle$

**fun** *output-sequences* :: ('x,'y) language  $\Rightarrow$  ('x,'y) word  $\Rightarrow$  'x list  $\Rightarrow$  'y list set  
**where**  
*output-sequences* L  $\pi$  xs = *output-projection* ' { $\tau \in \mathcal{L}[L,\pi] . [\tau]_I = xs$ }

**lemma** *prefix-closure-no-member* :  
**assumes** *is-language* X Y L  
**and**  $\pi \notin L$   
**shows**  $\pi @_{\tau} \notin L$   
 $\langle \text{proof} \rangle$

**lemma** *output-sequences-empty-iff* :  
**assumes** *is-language*  $X Y L$   
**shows**  $(\text{output-sequences } L \pi xs = \{\}) = ((\pi \notin L) \vee (\neg \text{is-executable } L \pi xs))$   
 $\langle \text{proof} \rangle$

**fun** *outputs* ::  $('x, 'y) \text{ language} \Rightarrow ('x, 'y) \text{ word} \Rightarrow 'x \Rightarrow 'y \text{ set}$  **where**  
 $\text{outputs } L \pi x = \{y . [(x, y)] \in \mathcal{L}[L, \pi]\}$

**notation** *outputs* ( $\langle \text{out}[-, -, -] \rangle$ )

**lemma** *outputs-in-alphabet* :  
**assumes** *is-language*  $X Y L$   
**shows**  $\text{out}[L, \pi, x] \subseteq Y$   
 $\langle \text{proof} \rangle$

**lemma** *outputs-executable* :  $(\text{out}[L, \pi, x] = \{\}) \longleftrightarrow (x \notin \text{exec}[L, \pi])$   
 $\langle \text{proof} \rangle$

**fun** *is-completely-specified-for* ::  $'x \text{ set} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$  **where**  
 $\text{is-completely-specified-for } X L = (\forall \pi \in L . \forall x \in X . \text{out}[L, \pi, x] \neq \{\})$

**lemma** *prefix-executable* :  
**assumes** *is-language*  $X Y L$   
**and**  $\pi \in L$   
**and**  $i < \text{length } \pi$   
**shows**  $\text{fst } (\pi ! i) \in \text{exec}[L, \text{take } i \pi]$   
 $\langle \text{proof} \rangle$

## 2 Conformance Relations

**definition** *language-equivalence* ::  $('x, 'y) \text{ language} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$   
**where**  
 $\text{language-equivalence } L1 L2 = (L1 = L2)$

**definition** *language-inclusion* ::  $('x, 'y) \text{ language} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$  **where**  
 $\text{language-inclusion } L1 L2 = (L1 \subseteq L2)$

**abbreviation**(*input*) *reduction*  $L1 L2 \equiv \text{language-inclusion } L1 L2$

**definition** *quasi-equivalence* ::  $('x, 'y) \text{ language} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$  **where**  
 $\text{quasi-equivalence } L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x \in \text{exec}[L2, \pi] . \text{out}[L1, \pi, x] =$

$out[L2,\pi,x])$

**definition** *quasi-reduction*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*quasi-reduction*  $L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x \in \text{exec}[L2,\pi] . (out[L1,\pi,x] \neq \{\} \wedge out[L1,\pi,x] \subseteq out[L2,\pi,x]))$

**definition** *strong-reduction*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*strong-reduction*  $L1 L2 = (\text{quasi-reduction } L1 L2 \wedge (\forall \pi \in L1 \cap L2 . \forall x . out[L2,\pi,x] = \{\} \longrightarrow out[L1,\pi,x] = \{\}))$

**definition** *semi-equivalence*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*semi-equivalence*  $L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x \in \text{exec}[L2,\pi] . (out[L1,\pi,x] = \{\} \vee out[L1,\pi,x] = out[L2,\pi,x]) \wedge (\exists x' . out[L1,\pi,x'] \cap out[L2,\pi,x'] \neq \{\}))$

**definition** *semi-reduction*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*semi-reduction*  $L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x \in \text{exec}[L2,\pi] . (out[L1,\pi,x] \subseteq out[L2,\pi,x]) \wedge (\exists x' . out[L1,\pi,x'] \cap out[L2,\pi,x'] \neq \{\}))$

**definition** *strong-semi-equivalence*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*strong-semi-equivalence*  $L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x . (x \in \text{exec}[L2,\pi] \longrightarrow ((out[L1,\pi,x] = \{\} \vee out[L1,\pi,x] = out[L2,\pi,x]) \wedge (\exists x' . out[L1,\pi,x'] \cap out[L2,\pi,x'] \neq \{\}))) \wedge (x \notin \text{exec}[L2,\pi] \longrightarrow out[L1,\pi,x] = \{\}))$

**definition** *strong-semi-reduction*  $:: ('x,'y) \text{ language} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*strong-semi-reduction*  $L1 L2 = (\forall \pi \in L1 \cap L2 . \forall x . (x \in \text{exec}[L2,\pi] \longrightarrow (out[L1,\pi,x] \subseteq out[L2,\pi,x] \wedge (\exists x' . out[L1,\pi,x'] \cap out[L2,\pi,x'] \neq \{\}))) \wedge (x \notin \text{exec}[L2,\pi] \longrightarrow out[L1,\pi,x] = \{\}))$

### 3 Unifying Characterisations

#### 3.1 $\preceq$ Conformance

**fun** *type-1-conforms*  $:: ('x,'y) \text{ language} \Rightarrow 'x \text{ alphabet} \Rightarrow 'y \text{ output-relation} \Rightarrow ('x,'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*type-1-conforms*  $L1 X H L2 = (\forall \pi \in L1 \cap L2 . \forall x \in X . (out[L1,\pi,x], out[L2,\pi,x] \in H))$

**notation** *type-1-conforms*  $(\leftarrow \preceq[-,-] \rightarrow)$

**fun** *equiv*  $:: 'y \text{ alphabet} \Rightarrow 'y \text{ output-relation}$  **where**  
*equiv*  $Y = \{(A,A) \mid A . A \subseteq Y\}$

**fun** *red*  $:: 'y \text{ alphabet} \Rightarrow 'y \text{ output-relation}$  **where**

$red\ Y = \{(A,B) \mid A\ B . A \subseteq B \wedge B \subseteq Y\}$

**fun** *quasieq* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*quasieq*  $Y = \{(A,A) \mid A . A \subseteq Y\} \cup \{(A,\{\}) \mid A . A \subseteq Y\}$

**fun** *quasired* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*quasired*  $Y = \{(A,B) \mid A\ B . A \neq \{\} \wedge A \subseteq B \wedge B \subseteq Y\} \cup \{(C,\{\}) \mid C . C \subseteq Y\}$

**fun** *strongred* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*strongred*  $Y = \{(A,B) \mid A\ B . A \neq \{\} \wedge A \subseteq B \wedge B \subseteq Y\} \cup \{(\{\},\{\})\}$

**lemma** *red-type-1* :  
  **assumes** *is-language*  $X\ Y\ L1$   
  **and** *is-language*  $X\ Y\ L2$   
**shows** *reduction*  $L1\ L2 \longleftrightarrow (L1 \preceq[X,red\ Y]\ L2)$   
   $\langle proof \rangle$

**lemma** *equiv-by-reduction* :  $(L1 \preceq[X,equiv\ Y]\ L2) \longleftrightarrow ((L1 \preceq[X,red\ Y]\ L2) \wedge (L2 \preceq[X,red\ Y]\ L1))$   
   $\langle proof \rangle$

**lemma** *equiv-type-1* :  
  **assumes** *is-language*  $X\ Y\ L1$   
  **and** *is-language*  $X\ Y\ L2$   
**shows**  $(L1 = L2) \longleftrightarrow (L1 \preceq[X,equiv\ Y]\ L2)$   
   $\langle proof \rangle$

**lemma** *quasired-type-1* :  
  **assumes** *is-language*  $X\ Y\ L1$   
  **and** *is-language*  $X\ Y\ L2$   
**shows** *quasi-reduction*  $L1\ L2 \longleftrightarrow (L1 \preceq[X,quasired\ Y]\ L2)$   
   $\langle proof \rangle$

**lemma** *quasieq-type-1* :  
  **assumes** *is-language*  $X\ Y\ L1$   
  **and** *is-language*  $X\ Y\ L2$   
**shows** *quasi-equivalence*  $L1\ L2 \longleftrightarrow (L1 \preceq[X,quasieq\ Y]\ L2)$   
   $\langle proof \rangle$

**lemma** *strongred-type-1* :  
  **assumes** *is-language*  $X\ Y\ L1$   
  **and** *is-language*  $X\ Y\ L2$

**shows** *strong-reduction*  $L1\ L2 \longleftrightarrow (L1 \preceq[X, \text{strongred } Y] L2)$   
 ⟨*proof*⟩

### 3.2 $\leq$ Conformance

**fun** *type-2-conforms* :: ('x,'y) language  $\Rightarrow$  'x alphabet  $\Rightarrow$  'y output-relation  $\Rightarrow$  ('x,'y) language  $\Rightarrow$  bool **where**  
*type-2-conforms*  $L1\ X\ H\ L2 =$  (  
 ( $\forall \pi \in L1 \cap L2 . \forall x \in X . (\text{out}[L1,\pi,x], \text{out}[L2,\pi,x] \in H) \wedge$   
 ( $\forall \pi \in L1 \cap L2 . \text{exec}[L2,\pi] \neq \{\} \longrightarrow (\exists x . \text{out}[L1,\pi,x] \cap \text{out}[L2,\pi,x] \neq \{\})$ )))

**notation** *type-2-conforms* ( $\langle \cdot \preceq[-,\cdot] \cdot \rangle$ )

**fun** *semieq* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*semieq*  $Y = \{(A,A) \mid A . A \subseteq Y\} \cup \{(\{\},A) \mid A . A \subseteq Y\} \cup \{(A,\{\}) \mid A . A \subseteq Y\}$

**fun** *semired* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*semired*  $Y = \{(A,B) \mid A\ B . A \subseteq B \wedge B \subseteq Y\} \cup \{(C,\{\}) \mid C . C \subseteq Y\}$

**fun** *strongsemieq* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*strongsemieq*  $Y = \{(A,A) \mid A . A \subseteq Y\} \cup \{(\{\},A) \mid A . A \subseteq Y\}$

**fun** *strongsemired* :: 'y alphabet  $\Rightarrow$  'y output-relation **where**  
*strongsemired*  $Y = \{(A,B) \mid A\ B . A \subseteq B \wedge B \subseteq Y\}$

**lemma** *strongsemieq-alt-def* : *strongsemieq*  $Y = \text{semieq } Y \cap \text{red } Y$   
 ⟨*proof*⟩

**lemma** *strongsemired-alt-def* : *strongsemired*  $Y = \text{red } Y$   
 ⟨*proof*⟩

**lemma** *semired-type-2* :  
**assumes** *is-language*  $X\ Y\ L1$   
**and** *is-language*  $X\ Y\ L2$   
**shows** (*semi-reduction*  $L1\ L2$ )  $\longleftrightarrow (L1 \leq[X, \text{semired } Y] L2)$   
 ⟨*proof*⟩

**lemma** *semieq-type-2* :  
**assumes** *is-language*  $X\ Y\ L1$   
**and** *is-language*  $X\ Y\ L2$   
**shows** (*semi-equivalence*  $L1\ L2$ )  $\longleftrightarrow (L1 \leq[X, \text{semieq } Y] L2)$   
 ⟨*proof*⟩

**lemma** *strongsemired-type-2* :

**assumes** *is-language*  $X Y L1$   
**and** *is-language*  $X Y L2$   
**shows** (*strong-semi-reduction*  $L1 L2$ )  $\longleftrightarrow$  ( $L1 \leq[X, \text{strongsemired } Y] L2$ )  
*<proof>*

**lemma** *strongsemieq-type-2* :  
**assumes** *is-language*  $X Y L1$   
**and** *is-language*  $X Y L2$   
**shows** (*strong-semi-equivalence*  $L1 L2$ )  $\longleftrightarrow$  ( $L1 \leq[X, \text{strongsemieq } Y] L2$ )  
*<proof>*

## 4 Comparing Conformance Relations

**lemma** *type-1-subset* :  
**assumes**  $L1 \preceq[X, H1] L2$   
**and**  $H1 \subseteq H2$   
**shows**  $L1 \preceq[X, H2] L2$   
*<proof>*

**lemma** *type-1-subsets* :  
**shows** *equiv*  $Y \subseteq$  *strongred*  $Y$   
**and** *equiv*  $Y \subseteq$  *quasieq*  $Y$   
**and** *strongred*  $Y \subseteq$  *red*  $Y$   
**and** *strongred*  $Y \subseteq$  *quasired*  $Y$   
**and** *quasieq*  $Y \subseteq$  *quasired*  $Y$   
*<proof>*

**lemma** *type-1-implications* :  
**shows**  $L1 \preceq[X, \text{equiv } Y] L2 \implies L1 \preceq[X, \text{strongred } Y] L2$   
**and**  $L1 \preceq[X, \text{equiv } Y] L2 \implies L1 \preceq[X, \text{red } Y] L2$   
**and**  $L1 \preceq[X, \text{equiv } Y] L2 \implies L1 \preceq[X, \text{quasired } Y] L2$   
**and**  $L1 \preceq[X, \text{equiv } Y] L2 \implies L1 \preceq[X, \text{quasieq } Y] L2$   
**and**  $L1 \preceq[X, \text{strongred } Y] L2 \implies L1 \preceq[X, \text{red } Y] L2$   
**and**  $L1 \preceq[X, \text{strongred } Y] L2 \implies L1 \preceq[X, \text{quasired } Y] L2$   
**and**  $L1 \preceq[X, \text{quasieq } Y] L2 \implies L1 \preceq[X, \text{quasired } Y] L2$   
*<proof>*

**lemma** *type-2-subset* :  
**assumes**  $L1 \leq[X, H1] L2$   
**and**  $H1 \subseteq H2$   
**shows**  $L1 \leq[X, H2] L2$   
*<proof>*

**lemma** *type-2-subsets* :  
**shows** *strongsemieq*  $Y \subseteq$  *strongsemired*  $Y$   
**and** *strongsemieq*  $Y \subseteq$  *semieq*  $Y$   
**and** *semieq*  $Y \subseteq$  *semired*  $Y$



**and**  $\text{strongsemired } Y \subseteq \text{semired } Y$   
**and**  $\text{strongsemired } Y \subseteq \text{red } Y$   
 ⟨proof⟩

**lemma** *type-2-implications* :

**shows**  $L1 \leq [X, \text{strongsemieq } Y] L2 \implies L1 \leq [X, \text{strongsemired } Y] L2$   
**and**  $L1 \leq [X, \text{strongsemieq } Y] L2 \implies L1 \leq [X, \text{semieq } Y] L2$   
**and**  $L1 \leq [X, \text{strongsemieq } Y] L2 \implies L1 \leq [X, \text{semired } Y] L2$   
**and**  $L1 \leq [X, \text{strongsemired } Y] L2 \implies L1 \leq [X, \text{semired } Y] L2$   
**and**  $L1 \leq [X, \text{semieq } Y] L2 \implies L1 \leq [X, \text{semired } Y] L2$   
 ⟨proof⟩

**lemma** *type-1-conformance-to-type-2* :

**assumes** *is-language*  $X Y L2$   
**and**  $L1 \preceq [X, H1] L2$   
**and**  $H1 \subseteq H2$   
**and**  $\bigwedge A B . (A, B) \in H1 \implies B \neq \{\} \implies A \cap B \neq \{\}$   
**shows**  $L1 \leq [X, H2] L2$   
 ⟨proof⟩

**lemma** *type-1-and-2-mixed-implications* :

**assumes** *is-language*  $X Y L2$   
**shows**  $L1 \leq [X, \text{strongsemieq } Y] L2 \implies L1 \preceq [X, \text{red } Y] L2$   
**and**  $L1 \leq [X, \text{strongsemired } Y] L2 \implies L1 \preceq [X, \text{red } Y] L2$   
**and**  $L1 \preceq [X, \text{quasieq } Y] L2 \implies L1 \leq [X, \text{semieq } Y] L2$   
**and**  $L1 \preceq [X, \text{quasired } Y] L2 \implies L1 \leq [X, \text{semired } Y] L2$   
**and**  $L1 \preceq [X, \text{equiv } Y] L2 \implies L1 \leq [X, \text{strongsemieq } Y] L2$   
**and**  $L1 \preceq [X, \text{strongred } Y] L2 \implies L1 \leq [X, \text{strongsemired } Y] L2$   
 ⟨proof⟩

## 4.1 Completely Specified Languages

**definition** *partiality-component* :: 'y set  $\implies$  'y output-relation **where**  
*partiality-component*  $Y = \{(A, \{\}) \mid A . A \subseteq Y\} \cup \{(\{\}, A) \mid A . A \subseteq Y\}$

**abbreviation**  $(\text{input}) \Pi Y \equiv \text{partiality-component } Y$

**lemma** *conformance-without-partiality* :

**shows**  $\text{strongsemieq } Y - \Pi Y = \text{semieq } Y - \Pi Y$   
**and**  $\text{semieq } Y - \Pi Y = \text{equiv } Y - \Pi Y$   
**and**  $\text{strongsemired } Y - \Pi Y = \text{semired } Y - \Pi Y$   
**and**  $\text{semired } Y - \Pi Y = \text{red } Y - \Pi Y$   
 ⟨proof⟩

## 5 Conformance Testing

**type-synonym**  $(\text{'x, 'y}) \text{state-cover} = (\text{'x, 'y}) \text{language}$

**type-synonym**  $(x,y)$  *transition-cover* =  $(x,y)$  *state-cover*  $\times$   $x$  *set*

**fun** *is-state-cover* ::  $(x,y)$  *language*  $\Rightarrow$   $(x,y)$  *language*  $\Rightarrow$   $(x,y)$  *state-cover*  $\Rightarrow$  *bool* **where**  
*is-state-cover*  $L1\ L2\ V = (\forall \pi \in L1 \cap L2 . \exists \alpha \in V . \mathcal{L}[L1,\pi] = \mathcal{L}[L1,\alpha] \wedge \mathcal{L}[L2,\pi] = \mathcal{L}[L2,\alpha])$

**lemma** *state-cover-subset* :  
**assumes** *is-language*  $X\ Y\ L1$   
**and** *is-language*  $X\ Y\ L2$   
**and** *is-state-cover*  $L1\ L2\ V$   
**and**  $\pi \in L1 \cap L2$   
**obtains**  $\alpha$  **where**  $\alpha \in V$   
**and**  $\alpha \in L1 \cap L2$   
**and**  $\mathcal{L}[L1,\pi] = \mathcal{L}[L1,\alpha]$   
**and**  $\mathcal{L}[L2,\pi] = \mathcal{L}[L2,\alpha]$   
 $\langle$ *proof* $\rangle$

**theorem** *sufficient-condition-for-type-1-conformance* :  
**assumes** *is-language*  $X\ Y\ L1$   
**and** *is-language*  $X\ Y\ L2$   
**and** *is-state-cover*  $L1\ L2\ V$   
**shows**  $(L1 \preceq[X,H] L2) \iff (\forall \pi \in V . \forall x \in X . \pi \in L1 \cap L2 \longrightarrow (out[L1,\pi,x], out[L2,\pi,x]) \in H)$   
 $\langle$ *proof* $\rangle$

**theorem** *sufficient-condition-for-type-2-conformance* :  
**assumes** *is-language*  $X\ Y\ L1$   
**and** *is-language*  $X\ Y\ L2$   
**and** *is-state-cover*  $L1\ L2\ V$   
**shows**  $(L1 \preceq[X,H] L2) \iff (\forall \pi \in V . \forall x \in X . \pi \in L1 \cap L2 \longrightarrow (out[L1,\pi,x], out[L2,\pi,x]) \in H \wedge (out[L2,\pi,x] \neq \{\} \longrightarrow (\exists x' \in X . out[L1,\pi,x'] \cap out[L2,\pi,x'] \neq \{\})))$   
 $\langle$ *proof* $\rangle$

**lemma** *intersections-card-helper* :  
**assumes** *finite*  $X$   
**and** *finite*  $Y$   
**shows**  $card \{A \cap B \mid A\ B . A \in X \wedge B \in Y\} \leq card\ X * card\ Y$   
 $\langle$ *proof* $\rangle$

**lemma** *prefix-length-take* :  
 $(prefix\ xs\ ys \wedge length\ xs \leq k) \iff (prefix\ xs\ (take\ k\ ys))$   
 $\langle$ *proof* $\rangle$

**lemma** *brute-force-state-cover* :  
**assumes** *is-language*  $X Y L1$   
**and** *is-language*  $X Y L2$   
**and** *finite*  $\{\mathcal{L}[L1,\pi] \mid \pi . \pi \in L1\}$   
**and** *finite*  $\{\mathcal{L}[L2,\pi] \mid \pi . \pi \in L2\}$   
**and** *card*  $\{\mathcal{L}[L1,\pi] \mid \pi . \pi \in L1\} \leq n$   
**and** *card*  $\{\mathcal{L}[L2,\pi] \mid \pi . \pi \in L2\} \leq m$   
**shows** *is-state-cover*  $L1 L2 \{\alpha . \text{length } \alpha \leq m * n - 1 \wedge (\forall xy \in \text{set } \alpha . \text{fst } xy \in X \wedge \text{snd } xy \in Y)\}$   
 $\langle \text{proof} \rangle$

## 6 Reductions Between Relations

### 6.1 Quasi-Equivalence via Quasi-Reduction and Absences

**fun** *absence-completion* ::  $'x \text{ alphabet} \Rightarrow 'y \text{ alphabet} \Rightarrow ('x, 'y) \text{ language} \Rightarrow ('x, 'y \times \text{bool}) \text{ language}$  **where**  
*absence-completion*  $X Y L =$   
 $((\lambda \pi . \text{map } (\lambda(x,y) . (x,(y, \text{True}))) \pi) ' L$   
 $\cup \{(\text{map } (\lambda(x,y) . (x,(y, \text{True}))) \pi) @ [(x,(y, \text{False}))] @ \tau \mid \pi x y \tau . \pi \in L \wedge$   
 $\text{out}[L,\pi,x] \neq \{\} \wedge y \in Y \wedge y \notin \text{out}[L,\pi,x] \wedge (\forall (x,(y,a)) \in \text{set } \tau . x \in X \wedge y \in Y)\}$

**lemma** *absence-completion-is-language* :  
**assumes** *is-language*  $X Y L$   
**shows** *is-language*  $X (Y \times \text{UNIV}) (\text{absence-completion } X Y L)$   
 $\langle \text{proof} \rangle$

**lemma** *absence-completion-inclusion-R* :  
**assumes** *is-language*  $X Y L$   
**and**  $\pi \in \text{absence-completion } X Y L$   
**shows**  $(\text{map } (\lambda(x,y,a) . (x,y)) \pi \in L) \longleftrightarrow (\forall (x,y,a) \in \text{set } \pi . a = \text{True})$   
 $\langle \text{proof} \rangle$

**lemma** *absence-completion-inclusion-L* :  
 $(\pi \in L) \longleftrightarrow (\text{map } (\lambda(x,y) . (x,y, \text{True})) \pi \in \text{absence-completion } X Y L)$   
 $\langle \text{proof} \rangle$

**fun** *is-present* ::  $('x, 'y \times \text{bool}) \text{ word} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$  **where**  
*is-present*  $\pi L = (\pi \in \text{map } (\lambda(x, y) . (x, y, \text{True})) ' L)$

**lemma** *is-present-rev* :  
**assumes** *is-present*  $\pi L$   
**shows**  $\text{map } (\lambda(x, y, a) . (x, y)) \pi \in L$   
 $\langle \text{proof} \rangle$

**lemma** *absence-completion-out* :

**assumes** *is-language*  $X Y L$

**and**  $x \in X$

**and**  $\pi \in \text{absence-completion } X Y L$

**shows**  $\text{is-present } \pi L \implies \text{out}[L, \text{map } (\lambda(x, y, a). (x, y)) \pi, x] \neq \{\} \implies \text{out}[\text{absence-completion } X Y L, \pi, x] = \{(y, \text{True}) \mid y \cdot y \in \text{out}[L, \text{map } (\lambda(x, y, a). (x, y)) \pi, x]\} \cup \{(y, \text{False}) \mid y \cdot y \in Y \wedge y \notin \text{out}[L, \text{map } (\lambda(x, y, a). (x, y)) \pi, x]\}$

**and**  $\text{is-present } \pi L \implies \text{out}[L, \text{map } (\lambda(x, y, a). (x, y)) \pi, x] = \{\} \implies \text{out}[\text{absence-completion } X Y L, \pi, x] = \{\}$

**and**  $\neg \text{is-present } \pi L \implies \text{out}[\text{absence-completion } X Y L, \pi, x] = Y \times \text{UNIV}$

*<proof>*

**theorem** *quasieq-via-quasired* :

**assumes** *is-language*  $X Y L1$

**and** *is-language*  $X Y L2$

**shows**  $(L1 \preceq[X, \text{quasieq } Y] L2) \longleftrightarrow ((\text{absence-completion } X Y L1) \preceq[X, \text{quasired } (Y \times \text{UNIV})] (\text{absence-completion } X Y L2))$

*<proof>*

## 6.2 Quasi-Reduction via Reduction and explicit Undefined Behaviour

**fun** *bottom-completion* ::  $'x \text{ alphabet} \Rightarrow 'y \text{ alphabet} \Rightarrow ('x, 'y) \text{ language} \Rightarrow ('x, 'y) \text{ option} \text{ language}$  **where**

*bottom-completion*  $X Y L =$

$((\lambda \pi \cdot \text{map } (\lambda(x, y). (x, \text{Some } y)) \pi) ' L)$

$\cup \{(\text{map } (\lambda(x, y). (x, \text{Some } y)) \pi) @ [(x, y)] @ \tau \mid \pi x y \tau \cdot \pi \in L \wedge \text{out}[L, \pi, x] = \{\} \wedge x \in X \wedge (y = \text{None} \vee y \in \text{Some } ' Y) \wedge (\forall (x, y) \in \text{set } \tau \cdot x \in X \wedge (y = \text{None} \vee y \in \text{Some } ' Y))\}$

**lemma** *bottom-completion-is-language* :

**assumes** *is-language*  $X Y L$

**shows** *is-language*  $X (\{\text{None}\} \cup \text{Some } ' Y)$  (*bottom-completion*  $X Y L$ )

*<proof>*

**fun** *is-not-undefined* ::  $('x, 'y) \text{ option} \text{ word} \Rightarrow ('x, 'y) \text{ language} \Rightarrow \text{bool}$  **where**

*is-not-undefined*  $\pi L = (\pi \in \text{map } (\lambda(x, y). (x, \text{Some } y)) ' L)$

**lemma** *bottom-id* :  $\text{map } (\lambda(x, y). (x, \text{the } y)) (\text{map } (\lambda(x, y). (x, \text{Some } y)) \pi) = \pi$

*<proof>*

**fun** *maximum-prefix-with-property* :: ('a list  $\Rightarrow$  bool)  $\Rightarrow$  'a list  $\Rightarrow$  'a list **where**  
*maximum-prefix-with-property* P xs = (last (filter P (prefixes xs)))

**lemma** *maximum-prefix-with-property-props* :  
**assumes**  $\exists$  ys  $\in$  set (prefixes xs) . P ys  
**shows** P (*maximum-prefix-with-property* P xs)  
**and** (*maximum-prefix-with-property* P xs)  $\in$  set (prefixes xs)  
**and**  $\bigwedge$  ys . prefix ys xs  $\Rightarrow$  P ys  $\Rightarrow$  length ys  $\leq$  length (*maximum-prefix-with-property* P xs)  
 <proof>

**lemma** *bottom-completion-out* :  
**assumes** is-language X Y L  
**and**  $x \in X$   
**and**  $\pi \in$  bottom-completion X Y L  
**shows** is-not-undefined  $\pi$  L  $\Rightarrow$  out[L, map ( $\lambda(x,y) . (x, \text{the } y)$ )  $\pi, x$ ]  $\neq$  {}  $\Rightarrow$   
 out[bottom-completion X Y L,  $\pi, x$ ] = Some ' out[L, map ( $\lambda(x,y) . (x, \text{the } y)$ )  $\pi, x$ ]  
**and** is-not-undefined  $\pi$  L  $\Rightarrow$  out[L, map ( $\lambda(x,y) . (x, \text{the } y)$ )  $\pi, x$ ] = {}  $\Rightarrow$   
 out[bottom-completion X Y L,  $\pi, x$ ] = {None}  $\cup$  Some ' Y  
**and**  $\neg$  is-not-undefined  $\pi$  L  $\Rightarrow$  out[bottom-completion X Y L,  $\pi, x$ ] = {None}  
 $\cup$  Some ' Y  
 <proof>

**theorem** *quasired-via-red* :  
**assumes** is-language X Y L1  
**and** is-language X Y L2  
**shows** (L1  $\preceq$ [X, quasired Y] L2)  $\longleftrightarrow$  ((bottom-completion X Y L1)  $\preceq$ [X, red  
 ({None}  $\cup$  Some ' Y)] (bottom-completion X Y L2))  
 <proof>

### 6.3 Strong Reduction via Reduction and Undefinedness Outputs

**fun** *non-bottom-shortening* :: ('x, 'y option) word  $\Rightarrow$  ('x, 'y option) word **where**  
*non-bottom-shortening*  $\pi$  = filter ( $\lambda(x,y) . y \neq \text{None}$ )  $\pi$

**fun** *non-bottom-projection* :: ('x, 'y option) word  $\Rightarrow$  ('x, 'y) word **where**  
*non-bottom-projection*  $\pi$  = map ( $\lambda(x,y) . (x, \text{the } y)$ ) (*non-bottom-shortening*  $\pi$ )

**lemma** *non-bottom-projection-split*: *non-bottom-projection* ( $\pi' @ \pi''$ ) = (*non-bottom-projection*  
 $\pi'$ ) @ (*non-bottom-projection*  $\pi''$ )  
 <proof>

**lemma** *non-bottom-projection-id* : *non-bottom-projection* (map ( $\lambda(x,y) . (x, \text{Some } y)$ )  $\pi$ ) =  $\pi$

*<proof>*

**fun** *undefinedness-completion* :: 'x alphabet  $\Rightarrow$  ('x,'y) language  $\Rightarrow$  ('x, 'y option) language **where**  
  *undefinedness-completion* X L =  
  { $\pi$  . *non-bottom-projection*  $\pi \in L \wedge (\forall \pi' x \pi'' . \pi = \pi' @ [(x, None)] @ \pi'' \rightarrow x \in X \wedge \text{out}[L, \text{non-bottom-projection } \pi', x] = \{\})$ }

**lemma** *undefinedness-completion-is-language* :  
  **assumes** *is-language* X Y L  
  **shows** *is-language* X ( $\{None\} \cup \text{Some } ' Y$ ) (*undefinedness-completion* X L)  
*<proof>*

**lemma** *undefinedness-completion-inclusion* :  
  **assumes**  $\pi \in L$   
  **shows** *map* ( $\lambda(x,y) . (x, \text{Some } y)$ )  $\pi \in \text{undefinedness-completion } X L$   
*<proof>*

**lemma** *undefinedness-completion-out-shortening* :  
  **assumes** *is-language* X Y L  
  **and**  $\pi \in \text{undefinedness-completion } X L$   
  **and**  $x \in X$   
  **shows**  $\text{out}[\text{undefinedness-completion } X L, \pi, x] = \text{out}[\text{undefinedness-completion } X L, \text{non-bottom-shortening } \pi, x]$   
*<proof>*

**lemma** *undefinedness-completion-out-projection-not-empty* :  
  **assumes** *is-language* X Y L  
  **and**  $\pi \in \text{undefinedness-completion } X L$   
  **and**  $x \in X$   
  **and**  $\text{out}[L, \text{non-bottom-projection } \pi, x] \neq \{\}$   
  **shows**  $\text{out}[\text{undefinedness-completion } X L, \text{non-bottom-shortening } \pi, x] = \text{Some } ' \text{out}[L, \text{non-bottom-projection } \pi, x]$   
*<proof>*

**lemma** *undefinedness-completion-out-projection-empty* :  
  **assumes** *is-language* X Y L  
  **and**  $\pi \in \text{undefinedness-completion } X L$   
  **and**  $x \in X$   
  **and**  $\text{out}[L, \text{non-bottom-projection } \pi, x] = \{\}$   
  **shows**  $\text{out}[\text{undefinedness-completion } X L, \text{non-bottom-shortening } \pi, x] = \{None\}$   
*<proof>*

**theorem** *strongred-via-red* :  
**assumes** *is-language X Y L1*  
**and** *is-language X Y L2*  
**shows**  $(L1 \preceq[X, \text{strongred } Y] L2) \longleftrightarrow ((\text{undefinedness-completion } X L1) \preceq[X, \text{red } (\{None\} \cup \text{Some } Y)] (\text{undefinedness-completion } X L2))$   
 $\langle \text{proof} \rangle$

**end**

## References

- [1] W.-l. Huang and R. Sachtleben. *Conformance Relations Between Input/Output Languages*, pages 49–67. Springer Nature Switzerland, Cham, 2023.