

# Possibilistic Noninterference\*

Andrei Popescu

Johannes Hölzl

June 17, 2024

## Abstract

We formalize a wide variety of Volpano/Smith-style noninterference notions for a while language with parallel composition. We systematize and classify these notions according to compositionality w.r.t. the language constructs. Compositionality yields sound syntactic criteria (a.k.a. type systems) in a uniform way.

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Bisimilarity, abstractly</b>	<b>4</b>
<b>3</b>	<b>The programming language and its semantics</b>	<b>6</b>
3.1	Syntax and operational semantics . . . . .	6
3.2	Sublanguages . . . . .	12
<b>4</b>	<b>During-execution security</b>	<b>16</b>
4.1	Basic setting . . . . .	16
4.2	Compatibility and discreteness . . . . .	17
4.3	Terminating-interactive discreteness . . . . .	19
4.4	Self-isomorphism . . . . .	20
4.5	MustT-interactive self-isomorphism . . . . .	21
4.6	Notions of bisimilarity . . . . .	22
<b>5</b>	<b>Compositionality of the during-execution security notions</b>	<b>37</b>
5.1	Discreteness versus language constructs: . . . . .	37
5.2	Discreteness versus language constructs: . . . . .	37
5.3	Self-Isomorphism versus language constructs: . . . . .	38
5.4	Self-Isomorphism versus language constructs: . . . . .	38
5.5	Strong bisimilarity versus language constructs . . . . .	39

---

\*Supported by the DFG project Ni 491/13-1 (part of the DFG priority program RS3) and the DFG RTG 1480.

5.5.1	01T-bisimilarity versus language constructs . . . . .	41
5.5.2	01-bisimilarity versus language constructs . . . . .	43
5.5.3	WT-bisimilarity versus language constructs . . . . .	47
5.5.4	T-bisimilarity versus language constructs . . . . .	49
5.5.5	W-bisimilarity versus language constructs . . . . .	51
<b>6</b>	<b>After-execution security</b>	<b>61</b>
6.1	Setup for bisimilarities . . . . .	61
6.2	Execution traces . . . . .	66
6.3	Relationship between during-execution and after-execution security . . . . .	67
<b>7</b>	<b>Concrete setting</b>	<b>68</b>
7.1	Programs from EXAMPLE 1 . . . . .	71

## 1 Introduction

This is a formalization of the mathematical development presented in the paper [1]:

- a uniform framework where a wide range of language-based noninterference variants from the literature are expressed and compared w.r.t. their *contracts*: the strength of the security properties they ensure weighed against the harshness of the syntactic conditions they enforce;
- syntactic criteria for proving that a program has a specific noninterference property, using only compositionality, which captures uniformly several security type-system results from the literature and suggests a further improved type system.

There are two auxiliary theories:

- MyTactics, introducing a few customized tactics;
- Bisim, describing an abstract notion of bisimilarity relation, namely, the greatest symmetric relation that is a fixpoint of a monotonic operator—this shall be instantiated to several concrete bisimilarity later.

The main theories of the development (shown in Fig. 1) are organized similarly to the sectionwise structure of [1]:

Language\_Semantics corresponds to §2 in [1]. It introduces and customizes the syntax and small-step operational semantics of a while language with parallel composition, using notations very similar to the paper.

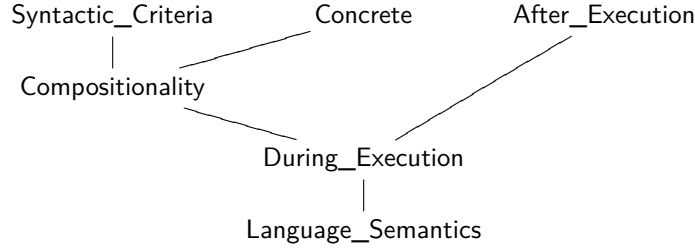


Figure 1: Main Theory Structure

`During_Execution`<sup>1</sup> mainly corresponds to §3 in [1], defining the various coinductive notions from there: self isomorphism, discreteness, variations of strong, weak and 01-bisimilarity. Prop. 1 from the paper, stating implications between these notions, is proved as the theorems `bis_imp` and `siso_bis`.<sup>2</sup> The bisimilarity inclusions stated in `bis_imp` are slightly more general than those in Prop. 1, in that they employ the binary version of the relation, e.g.,  $c \approx_s d \implies c \approx_{\text{WT}} d$  instead of  $c \approx_s c \implies c \approx_{\text{WT}} c$ .

`Compositionality` mainly corresponds to the homonymous §4 in [1]. The paper’s compositionality result, Prop. 2, is scattered through the theory as theorems with self-explanatory names, indicating the compositionality relationship between notions of noninterference and language constructs, e.g., `While_WbisT` (while versus termination-sensitive weak bisimilarity), `Par_ZO_bis` (parallel composition versus 01-bisimilarity).

Theories `During_Execution` and `Compositionality` also include the novel notion of noninterference  $\approx_{\tau}$  introduced in §5 of [1], based on the “must terminate” condition, which is given the same treatment as the other notions: `bis_imp` in `During_Execution` states the implication relationship between  $\approx_{\tau}$  and the other bisimilarities (Prop. 3.(1) from [1]), while various intuitively named theorems from `Language_Semantics` state the compositionality properties of  $\approx_{\tau}$  (Prop. 3.(2) from [1]).

`Syntactic_Criteria` corresponds to the homonymous §6 in [1]. The syntactic analogues of the semantics notions, indicated in the paper by overlining, e.g., `discr`, are in the scripts prefixed by “SC” (from “syntactic criterion”), e.g., `SC_discr`, `SC_WbisT`. Props. 4 and 5 from the paper (stating the relationship between the syntactic and the semantic notions and the implications between the syntactic notions, respectively) are again scattered through the theory under self-explanatory names.

`Concrete` contains an instantiation of the indistinguishability relation  $\sim$  from [1] to the standard two-level security setting described in the paper’s Exam-

<sup>1</sup>“During-execution” (bisimilarity-based) noninterference should be contrasted with “after-execution” (trace-based) noninterference according to the distinction made in [1] at the beginning of §7.

<sup>2</sup>To help the reader distinguish the main results from the auxiliary lemmas, the former are marked in the scripts with the keyword “theorem”.

ple 2.

Finally, `After_Execution` corresponds to §7 in [1], dealing with the after-execution guarantees of the during-execution notions of security. Prop. 6 in the paper is stated in the scripts as theorems `Sbis_trace`, `ZObisT_trace` and `WbisT_trace`, Prop. 7 as theorems `ZObis_trace` and `Wbis_trace`, and Prop. 8 as theorem `BisT_trace`.

## 2 Bisimilarity, abstractly

```
theory Bisim  
imports Interface  
begin
```

```
type-synonym 'a rel = ('a * 'a) set  
type-synonym ('cmd,'state) config = 'cmd * 'state
```

```
definition mono where  
mono Retr  $\equiv$   
 $\forall$  theta theta'. theta  $\leq$  theta'  $\longrightarrow$  Retr theta  $\leq$  Retr theta'
```

```
definition simul where  
simul Retr theta  $\equiv$  theta  $\leq$  Retr theta
```

```
definition bisim where  
bisim Retr theta  $\equiv$  sym theta  $\wedge$  simul Retr theta
```

```
lemma mono-Union:  
assumes mono Retr  
shows Union (Retr ' Theta)  $\leq$  Retr (Union Theta)  
{proof}
```

```
lemma mono-Un:  
assumes mono Retr  
shows Retr theta Un Retr theta'  $\subseteq$  Retr (theta Un theta')  
{proof}
```

```
lemma sym-Union:  
assumes  $\bigwedge$  theta. theta  $\in$  Theta  $\implies$  sym theta  
shows sym (Union Theta)  
{proof}
```

```
lemma sym-Un:  
assumes sym theta1 and sym theta2  
shows sym (theta1 Un theta2)  
{proof}
```

```
lemma simul-Union:  
assumes mono Retr
```

**and**  $\bigwedge \theta. \theta \in \Theta \implies \text{simul Retr } \theta$   
**shows**  $\text{simul Retr } (\text{Union } \Theta)$   
*<proof>*

**lemma** *simul-Un*:  
**assumes**  $\text{mono Retr}$  **and**  $\text{simul Retr } \theta_1$  **and**  $\text{simul Retr } \theta_2$   
**shows**  $\text{simul Retr } (\theta_1 \text{ Un } \theta_2)$   
*<proof>*

**lemma** *bisim-Union*:  
**assumes**  $\text{mono Retr}$  **and**  $\bigwedge \theta. \theta \in \Theta \implies \text{bisim Retr } \theta$   
**shows**  $\text{bisim Retr } (\text{Union } \Theta)$   
*<proof>*

**lemma** *bisim-Un*:  
**assumes**  $\text{mono Retr}$  **and**  $\text{bisim Retr } \theta_1$  **and**  $\text{bisim Retr } \theta_2$   
**shows**  $\text{bisim Retr } (\theta_1 \text{ Un } \theta_2)$   
*<proof>*

**definition** *bis where*  
 $\text{bis Retr} \equiv \text{Union } \{\theta. \text{bisim Retr } \theta\}$

**lemma** *bisim-bis[simp]*:  
**assumes**  $\text{mono Retr}$   
**shows**  $\text{bisim Retr } (\text{bis Retr})$   
*<proof>*

**corollary** *sym-bis[simp]*:  $\text{mono Retr} \implies \text{sym } (\text{bis Retr})$   
**and** *simul-bis[simp]*:  $\text{mono Retr} \implies \text{simul Retr } (\text{bis Retr})$   
*<proof>*

**lemma** *bis-raw-coind*:  
**assumes**  $\text{mono Retr}$  **and**  $\text{sym } \theta$  **and**  $\theta \subseteq \text{Retr } \theta$   
**shows**  $\theta \subseteq \text{bis Retr}$   
*<proof>*

**lemma** *bis-prefix[simp]*:  
**assumes**  $\text{mono Retr}$   
**shows**  $\text{bis Retr} \subseteq \text{Retr } (\text{bis Retr})$   
*<proof>*

**lemma** *bis-coind*:  
**assumes**  $*$ :  $\text{mono Retr}$  **and**  $\text{sym } \theta$  **and**  $*$ :  $\theta \subseteq \text{Retr } (\theta \text{ Un } (\text{bis Retr}))$   
**shows**  $\theta \subseteq \text{bis Retr}$   
*<proof>*

**lemma** *bis-coind2*:  
**assumes**  $*$ :  $\text{mono Retr}$  **and**  
 $*$ :  $\theta \subseteq \text{Retr } (\theta \text{ Un } (\text{bis Retr}))$  **and**

```

***:  $\theta^{-1} \subseteq \text{Retr} ((\theta^{-1}) \text{Un} (\text{bis Retr}))$ 
shows  $\theta \subseteq \text{bis Retr}$ 
⟨proof⟩

```

```

lemma bis-raw-coind2:
assumes *: mono Retr and
**:  $\theta \subseteq \text{Retr} \theta$  and
***:  $\theta^{-1} \subseteq \text{Retr} (\theta^{-1})$ 
shows  $\theta \subseteq \text{bis Retr}$ 
⟨proof⟩

```

```

lemma mono-bis:
assumes mono Retr1 and mono Retr2
and  $\bigwedge \theta. \text{Retr1 } \theta \subseteq \text{Retr2 } \theta$ 
shows  $\text{bis Retr1} \subseteq \text{bis Retr2}$ 
⟨proof⟩

```

**end**

### 3 The programming language and its semantics

**theory** *Language-Semantics* **imports** *Interface* **begin**

#### 3.1 Syntax and operational semantics

```

datatype ('test,'atom) com =
  Atm 'atom |
  Seq ('test,'atom) com ('test,'atom) com
  (- ;; - [60, 61] 60) |
  If 'test ('test,'atom) com ('test,'atom) com
  ((if -/ then -/ else -) [0, 0, 61] 61) |
  While 'test ('test,'atom) com
  ((while -/ do -) [0, 61] 61) |
  Par ('test,'atom) com ('test,'atom) com
  (- | - [60, 61] 60)

```

```

locale PL =
fixes
  tval :: 'test  $\Rightarrow$  'state  $\Rightarrow$  bool and
  aval :: 'atom  $\Rightarrow$  'state  $\Rightarrow$  'state

```

```

context PL
begin

```

Conventions and notations: – suffixes: “C” for “Continuation”, “T” for “termination” – prefix: “M” for multistep – tst, tst’ are tests – atm, atm’ are atoms (atomic commands) – s, s’, t, t’ are states – c, c’, d, d’ are commands – cf, cf’ are configurations, i.e., pairs command-state

**inductive** *transT* ::  
 (('test,'atom)com \* 'state) ⇒ 'state ⇒ bool  
 (**infix** →t 55)  
**where**  
*Atm*[simp]:  
 (Atm atm, s) →t aval atm s  
 | *WhileFalse*[simp]:  
 ~ tval tst s ⇒ (While tst c, s) →t s

**lemmas** *trans-Atm* = *Atm*  
**lemmas** *trans-WhileFalse* = *WhileFalse*

**inductive** *transC* ::  
 (('test,'atom)com \* 'state) ⇒ (('test,'atom)com \* 'state) ⇒ bool  
 (**infix** →c 55)  
**and** *MtransC* ::  
 (('test,'atom)com \* 'state) ⇒ (('test,'atom)com \* 'state) ⇒ bool  
 (**infix** →\*c 55)  
**where**  
*SeqC*[simp]:  
 (c1, s) →c (c1', s') ⇒ (c1 ;; c2, s) →c (c1' ;; c2, s')  
 | *SeqT*[simp]:  
 (c1, s) →t s' ⇒ (c1 ;; c2, s) →c (c2, s')  
 | *IfTrue*[simp]:  
 tval tst s ⇒ (If tst c1 c2, s) →c (c1, s)  
 | *IfFalse*[simp]:  
 ~ tval tst s ⇒ (If tst c1 c2, s) →c (c2, s)  
 | *WhileTrue*[simp]:  
 tval tst s ⇒ (While tst c, s) →c (c ;; (While tst c), s)

| *ParCL*[simp]:  
 (c1, s) →c (c1', s') ⇒ (Par c1 c2, s) →c (Par c1' c2, s')  
 | *ParCR*[simp]:  
 (c2, s) →c (c2', s') ⇒ (Par c1 c2, s) →c (Par c1 c2', s')  
 | *ParTL*[simp]:  
 (c1, s) →t s' ⇒ (Par c1 c2, s) →c (c2, s')  
 | *ParTR*[simp]:  
 (c2, s) →t s' ⇒ (Par c1 c2, s) →c (c1, s')  
 | *Refl*:  
 (c,s) →\*c (c,s)  
 | *Step*:  
 [[(c,s) →\*c (c',s'); (c',s') →c (c'',s'')] ⇒ (c,s) →\*c (c'',s')

**lemmas** *trans-SeqC* = *SeqC* **lemmas** *trans-SeqT* = *SeqT*  
**lemmas** *trans-IfTrue* = *IfTrue* **lemmas** *trans-IfFalse* = *IfFalse*  
**lemmas** *trans-WhileTrue* = *WhileTrue*  
**lemmas** *trans-ParCL* = *ParCL* **lemmas** *trans-ParCR* = *ParCR*  
**lemmas** *trans-ParTL* = *ParTL* **lemmas** *trans-ParTR* = *ParTR*

**lemmas** *trans-Refl* = *Refl* **lemmas** *trans-Step* = *Step*

**lemma** *MtransC-Refl*[*simp*]:  $cf \rightarrow^*c cf$   
<*proof*>

**lemmas** *transC-induct* = *transC-MtransC.inducts*(1)  
[*split-format*(*complete*),

**where**  $?P2.0 = \lambda c s c' s'. True$ ]

**lemmas** *MtransC-induct-temp* = *transC-MtransC.inducts*(2)[*split-format*(*complete*)]

**inductive** *MtransT* ::

$((\text{'test}, \text{'atom})\text{com} * \text{'state}) \Rightarrow \text{'state} \Rightarrow \text{bool}$

(**infix**  $\rightarrow^*t$  55)

**where**

*StepT*:

$\llbracket cf \rightarrow^*c cf'; cf' \rightarrow t s'' \rrbracket \Longrightarrow cf \rightarrow^*t s''$

**lemma** *MtransC-rtranclp-transC*:

$MtransC = transC \hat{\ }^*$

<*proof*>

**lemma** *transC-MtransC*[*simp*]:

**assumes**  $cf \rightarrow c cf'$

**shows**  $cf \rightarrow^*c cf'$

<*proof*>

**lemma** *MtransC-Trans*:

**assumes**  $cf \rightarrow^*c cf'$  **and**  $cf' \rightarrow^*c cf''$

**shows**  $cf \rightarrow^*c cf''$

<*proof*>

**lemma** *MtransC-StepC*:

**assumes**  $*$ :  $cf \rightarrow^*c cf'$  **and**  $**$ :  $cf' \rightarrow c cf''$

**shows**  $cf \rightarrow^*c cf''$

<*proof*>

**lemma** *MtransC-induct*[*consumes 1*, *case-names Refl Trans*]:

**assumes**  $cf \rightarrow^*c cf'$

**and**  $\bigwedge cf. phi\ cf\ cf'$

**and**

$\bigwedge cf\ cf'\ cf''.$

$\llbracket cf \rightarrow^*c cf'; phi\ cf\ cf'; cf' \rightarrow c cf'' \rrbracket$

$\Longrightarrow phi\ cf\ cf''$

**shows**  $phi\ cf\ cf'$

<*proof*>

**lemma** *MtransC-induct2*[*consumes 1*, *case-names Refl Trans*, *induct pred: MtransC*]:

**assumes**  $(c,s) \rightarrow^*c (c',s')$

**and**  $\bigwedge c\ s. phi\ c\ s\ c\ s$



**and**

$\bigwedge c\ s\ c'\ s'\ c''\ s''.$   
 $\llbracket (c,s) \rightarrow^* c\ (c',s');\ \text{phi}\ c\ s\ c'\ s';\ (c',s') \rightarrow c\ (c'',s'') \rrbracket$   
 $\implies \text{phi}\ c\ s\ c''\ s''$   
**shows**  $\text{phi}\ c\ s\ c'\ s'$   
 $\langle \text{proof} \rangle$

**lemma** *transT-MtransT[simp]*:

**assumes**  $cf \rightarrow t\ s'$   
**shows**  $cf \rightarrow^* t\ s'$   
 $\langle \text{proof} \rangle$

**lemma** *MtransC-MtransT*:

**assumes**  $cf \rightarrow^* c\ cf'$  **and**  $cf' \rightarrow^* t\ cf''$   
**shows**  $cf \rightarrow^* t\ cf''$   
 $\langle \text{proof} \rangle$

**lemma** *transC-MtransT[simp]*:

**assumes**  $cf \rightarrow c\ cf'$  **and**  $cf' \rightarrow^* t\ s''$   
**shows**  $cf \rightarrow^* t\ s''$   
 $\langle \text{proof} \rangle$

Inversion rules, nchotomies and such:

**lemma** *Atm-transC-simp[simp]*:

$\sim (Atm\ atm,\ s) \rightarrow c\ cf$   
 $\langle \text{proof} \rangle$

**lemma** *Atm-transC-invert[elim!]*:

**assumes**  $(Atm\ atm,\ s) \rightarrow c\ cf$   
**shows**  $\text{phi}$   
 $\langle \text{proof} \rangle$

**lemma** *Atm-transT-invert[elim!]*:

**assumes**  $(Atm\ atm,\ s) \rightarrow t\ s'$   
**and**  $s' = \text{aval}\ atm\ s \implies \text{phi}$   
**shows**  $\text{phi}$   
 $\langle \text{proof} \rangle$

**lemma** *Seq-transC-invert[elim!]*:

**assumes**  $(c1\ ;;\ c2,\ s) \rightarrow c\ (c',\ s')$   
**and**  $\bigwedge c1'. \llbracket (c1,\ s) \rightarrow c\ (c1',s');\ c' = c1' ;;\ c2 \rrbracket \implies \text{phi}$   
**and**  $\llbracket (c1,\ s) \rightarrow t\ s';\ c' = c2 \rrbracket \implies \text{phi}$   
**shows**  $\text{phi}$   
 $\langle \text{proof} \rangle$

**lemma** *Seq-transT-invert[simp]*:

$\sim (c1\ ;;\ c2,\ s) \rightarrow t\ s'$   
 $\langle \text{proof} \rangle$

**lemma** *If-transC-invert[elim!]*:  
**assumes**  $(\text{If } \text{tst } c1 \ c2, s) \rightarrow c (c', s')$   
**and**  $\llbracket \text{tval } \text{tst } s; c' = c1; s' = s \rrbracket \Longrightarrow \text{phi}$   
**and**  $\llbracket \sim \text{tval } \text{tst } s; c' = c2; s' = s \rrbracket \Longrightarrow \text{phi}$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *If-transT-simp[simp]*:  
 $\sim (\text{If } b \ c1 \ c2, s) \rightarrow t \ s'$   
 $\langle \text{proof} \rangle$

**lemma** *If-transT-invert[elim!]*:  
**assumes**  $(\text{If } b \ c1 \ c2, s) \rightarrow t \ s'$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *While-transC-invert[elim]*:  
**assumes**  $(\text{While } \text{tst } c1, s) \rightarrow c (c', s')$   
**and**  $\llbracket \text{tval } \text{tst } s; c' = c1 ;; (\text{While } \text{tst } c1); s' = s \rrbracket \Longrightarrow \text{phi}$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *While-transT-invert[elim!]*:  
**assumes**  $(\text{While } \text{tst } c1, s) \rightarrow t \ s'$   
**and**  $\llbracket \sim \text{tval } \text{tst } s; s' = s \rrbracket \Longrightarrow \text{phi}$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *Par-transC-invert[elim]*:  
**assumes**  $(\text{Par } c1 \ c2, s) \rightarrow c (c', s')$   
**and**  $\bigwedge c1'. \llbracket (c1, s) \rightarrow c (c1', s'); c' = \text{Par } c1' \ c2 \rrbracket \Longrightarrow \text{phi}$   
**and**  $\llbracket (c1, s) \rightarrow t \ s'; c' = c2 \rrbracket \Longrightarrow \text{phi}$   
**and**  $\bigwedge c2'. \llbracket (c2, s) \rightarrow c (c2', s'); c' = \text{Par } c1 \ c2' \rrbracket \Longrightarrow \text{phi}$   
**and**  $\llbracket (c2, s) \rightarrow t \ s'; c' = c1 \rrbracket \Longrightarrow \text{phi}$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *Par-transT-simp[simp]*:  
 $\sim (\text{Par } c1 \ c2, s) \rightarrow t \ s'$   
 $\langle \text{proof} \rangle$

**lemma** *Par-transT-invert[elim]*:  
**assumes**  $(\text{Par } c1 \ c2, s) \rightarrow t \ s'$   
**shows** *phi*  
 $\langle \text{proof} \rangle$

**lemma** *trans-nchotomy*:  
 $(\exists c' \ s'. (c, s) \rightarrow c (c', s')) \vee$   
 $(\exists s'. (c, s) \rightarrow t \ s')$

$\langle proof \rangle$

**corollary** *trans-invert*:

**assumes**

$\bigwedge c' s'. (c, s) \rightarrow c (c', s') \implies phi$

**and**  $\bigwedge s'. (c, s) \rightarrow t s' \implies phi$

**shows** *phi*

$\langle proof \rangle$

**lemma** *not-transC-transT*:

$\llbracket cf \rightarrow c cf'; cf \rightarrow t s' \rrbracket \implies phi$

$\langle proof \rangle$

**lemmas** *MtransT-invert = MtransT.cases*

**lemma** *MtransT-invert2*:

**assumes**  $(c, s) \rightarrow *t s''$

**and**  $\bigwedge c' s'. \llbracket (c, s) \rightarrow *c (c', s'); (c', s') \rightarrow t s'' \rrbracket \implies phi$

**shows** *phi*

$\langle proof \rangle$

**lemma** *Seq-MtransC-invert[elim!]*:

**assumes**  $(c1 ;; c2, s) \rightarrow *c (d', t')$

**and**  $\bigwedge c1'. \llbracket (c1, s) \rightarrow *c (c1', t'); d' = c1' ;; c2 \rrbracket \implies phi$

**and**  $\bigwedge s'. \llbracket (c1, s) \rightarrow *t s'; (c2, s') \rightarrow *c (d', t') \rrbracket \implies phi$

**shows** *phi*

$\langle proof \rangle$

**lemma** *Seq-MtransT-invert[elim!]*:

**assumes**  $*(c1 ;; c2, s) \rightarrow *t s''$

**and**  $** \bigwedge s'. \llbracket (c1, s) \rightarrow *t s'; (c2, s') \rightarrow *t s'' \rrbracket \implies phi$

**shows** *phi*

$\langle proof \rangle$

Direct rules for the multi-step relations

**lemma** *Seq-MtransC[simp]*:

**assumes**  $(c1, s) \rightarrow *c (c1', s')$

**shows**  $(c1 ;; c2, s) \rightarrow *c (c1' ;; c2, s')$

$\langle proof \rangle$

**lemma** *Seq-MtransT-MtransC[simp]*:

**assumes**  $(c1, s) \rightarrow *t s'$

**shows**  $(c1 ;; c2, s) \rightarrow *c (c2, s')$

$\langle proof \rangle$

**lemma** *ParCL-MtransC[simp]*:

**assumes**  $(c1, s) \rightarrow *c (c1', s')$

**shows**  $(Par c1 c2, s) \rightarrow *c (Par c1' c2, s')$

$\langle proof \rangle$

**lemma** *ParCR-MtransC[simp]*:  
**assumes**  $(c2, s) \rightarrow^* c (c2', s')$   
**shows**  $(Par\ c1\ c2, s) \rightarrow^* c (Par\ c1\ c2', s')$   
 $\langle proof \rangle$

**lemma** *ParTL-MtransC[simp]*:  
**assumes**  $(c1, s) \rightarrow^* t s'$   
**shows**  $(Par\ c1\ c2, s) \rightarrow^* c (c2, s')$   
 $\langle proof \rangle$

**lemma** *ParTR-MtransC[simp]*:  
**assumes**  $(c2, s) \rightarrow^* t s'$   
**shows**  $(Par\ c1\ c2, s) \rightarrow^* c (c1, s')$   
 $\langle proof \rangle$

### 3.2 Sublanguages

**fun** *noWhile* **where**  
 $noWhile\ (Atm\ atm) = True$   
 $|noWhile\ (c1\ ;;\ c2) = (noWhile\ c1 \wedge noWhile\ c2)$   
 $|noWhile\ (If\ b\ c1\ c2) = (noWhile\ c1 \wedge noWhile\ c2)$   
 $|noWhile\ (While\ b\ c) = False$   
 $|noWhile\ (Par\ c1\ c2) = (noWhile\ c1 \wedge noWhile\ c2)$

**fun** *seq* **where**  
 $seq\ (Atm\ atm) = True$   
 $|seq\ (c1\ ;;\ c2) = (seq\ c1 \wedge seq\ c2)$   
 $|seq\ (If\ b\ c1\ c2) = (seq\ c1 \wedge seq\ c2)$   
 $|seq\ (While\ b\ c) = seq\ c$   
 $|seq\ (Par\ c1\ c2) = False$

**lemma** *noWhile-transC*:  
**assumes** *noWhile*  $c$  **and**  $(c, s) \rightarrow c (c', s')$   
**shows** *noWhile*  $c'$   
 $\langle proof \rangle$

**lemma** *seq-transC*:  
**assumes** *seq*  $c$  **and**  $(c, s) \rightarrow c (c', s')$   
**shows** *seq*  $c'$   
 $\langle proof \rangle$

**abbreviation** *wfP-on* **where**  
 $wfP\text{-on}\ phi\ A \equiv wfP\ (\lambda a\ b. a \in A \wedge b \in A \wedge phi\ a\ b)$

**fun** *numSt* **where**  
 $numSt\ (Atm\ atm) = Suc\ 0$

$|numSt (c1 ;; c2) = numSt c1 + numSt c2$   
 $|numSt (If b c1 c2) = 1 + max (numSt c1) (numSt c2)$   
 $|numSt (Par c1 c2) = numSt c1 + numSt c2$

**lemma** *numSt-gt-0[simp]*:  
*noWhile c*  $\implies numSt c > 0$   
 ⟨*proof*⟩

**lemma** *numSt-transC*:  
**assumes** *noWhile c* **and**  $(c,s) \rightarrow c (c',s')$   
**shows**  $numSt c' < numSt c$   
 ⟨*proof*⟩

**corollary** *wfP-tranC-noWhile*:  
 $wfP (\lambda (c',s') (c,s). noWhile c \wedge (c,s) \rightarrow c (c',s'))$   
 ⟨*proof*⟩

**lemma** *noWhile-MtransT*:  
**assumes** *noWhile c*  
**shows**  $\exists s'. (c,s) \rightarrow^* t s'$   
 ⟨*proof*⟩

**coinductive** *mayDiverge* **where**  
*intro*:  
 $\llbracket (c,s) \rightarrow c (c',s') \wedge mayDiverge c' s' \rrbracket$   
 $\implies mayDiverge c s$

Coinduction for may-diverge :

**lemma** *mayDiverge-coind[consumes 1, case-names Hyp, induct pred: mayDiverge]*:  
**assumes** \*: *phi c s* **and**  
 \*\*:  $\bigwedge c s. phi c s \implies$   
 $\exists c' s'. (c,s) \rightarrow c (c',s') \wedge (phi c' s' \vee mayDiverge c' s')$   
**shows**  $mayDiverge c s$   
 ⟨*proof*⟩

**lemma** *mayDiverge-raw-coind[consumes 1, case-names Hyp]*:  
**assumes** \*: *phi c s* **and**  
 \*\*:  $\bigwedge c s. phi c s \implies$   
 $\exists c' s'. (c,s) \rightarrow c (c',s') \wedge phi c' s'$   
**shows**  $mayDiverge c s$   
 ⟨*proof*⟩

May-diverge versus transition:

**lemma** *mayDiverge-transC*:  
**assumes**  $mayDiverge c s$   
**shows**  $\exists c' s'. (c,s) \rightarrow c (c',s') \wedge mayDiverge c' s'$   
 ⟨*proof*⟩

**lemma** *transC-mayDiverge*:  
**assumes**  $(c,s) \rightarrow c (c',s')$  **and** *mayDiverge*  $c' s'$   
**shows** *mayDiverge*  $c s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-not-transT*:  
**assumes** *mayDiverge*  $c s$   
**shows**  $\neg (c,s) \rightarrow t s'$   
 $\langle proof \rangle$

**lemma** *MtransC-mayDiverge*:  
**assumes**  $(c,s) \rightarrow *c (c',s')$  **and** *mayDiverge*  $c' s'$   
**shows** *mayDiverge*  $c s$   
 $\langle proof \rangle$

**lemma** *not-MtransT-mayDiverge*:  
**assumes**  $\bigwedge s'. \neg (c,s) \rightarrow *t s'$   
**shows** *mayDiverge*  $c s$   
 $\langle proof \rangle$

**lemma** *not-mayDiverge-Atm[simp]*:  
 $\neg \text{mayDiverge } (Atm\ atm)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-Seq-L*:  
**assumes** *mayDiverge*  $c1\ s$   
**shows** *mayDiverge*  $(c1\ ;;\ c2)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-Seq-R*:  
**assumes**  $c1: (c1, s) \rightarrow *t s'$  **and**  $c2: \text{mayDiverge } c2\ s'$   
**shows** *mayDiverge*  $(c1\ ;;\ c2)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-If-L*:  
**assumes** *tval tst s* **and** *mayDiverge*  $c1\ s$   
**shows** *mayDiverge*  $(If\ tst\ c1\ c2)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-If-R*:  
**assumes**  $\neg \text{tval } tst\ s$  **and** *mayDiverge*  $c2\ s$   
**shows** *mayDiverge*  $(If\ tst\ c1\ c2)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-If*:  
**assumes** *mayDiverge*  $c1\ s$  **and** *mayDiverge*  $c2\ s$   
**shows** *mayDiverge*  $(If\ tst\ c1\ c2)\ s$   
 $\langle proof \rangle$

**lemma** *mayDiverge-Par-L*:  
**assumes** *mayDiverge c1 s*  
**shows** *mayDiverge (Par c1 c2) s*  
*<proof>*

**lemma** *mayDiverge-Par-R*:  
**assumes** *mayDiverge c2 s*  
**shows** *mayDiverge (Par c1 c2) s*  
*<proof>*

**definition** *mustT* **where**  
*mustT c s*  $\equiv \neg$  *mayDiverge c s*

**lemma** *mustT-transC*:  
**assumes** *mustT c s* **and**  $(c,s) \rightarrow c (c',s')$   
**shows** *mustT c' s'*  
*<proof>*

**lemma** *transT-not-mustT*:  
**assumes**  $(c,s) \rightarrow t s'$   
**shows** *mustT c s*  
*<proof>*

**lemma** *mustT-MtransC*:  
**assumes** *mustT c s* **and**  $(c,s) \rightarrow^* c (c',s')$   
**shows** *mustT c' s'*  
*<proof>*

**lemma** *mustT-MtransT*:  
**assumes** *mustT c s*  
**shows**  $\exists s'. (c,s) \rightarrow^* t s'$   
*<proof>*

**lemma** *mustT-Atm[simp]*:  
*mustT (Atm atm) s*  
*<proof>*

**lemma** *mustT-Seq-L*:  
**assumes** *mustT (c1 ;; c2) s*  
**shows** *mustT c1 s*  
*<proof>*

**lemma** *mustT-Seq-R*:  
**assumes** *mustT (c1 ;; c2) s* **and**  $(c1, s) \rightarrow^* t s'$   
**shows** *mustT c2 s'*  
*<proof>*

**lemma** *mustT-If-L*:  
**assumes** *tval tst s* **and** *mustT (If tst c1 c2) s*  
**shows** *mustT c1 s*  
 $\langle$ *proof* $\rangle$

**lemma** *mustT-If-R*:  
**assumes**  $\neg$  *tval tst s* **and** *mustT (If tst c1 c2) s*  
**shows** *mustT c2 s*  
 $\langle$ *proof* $\rangle$

**lemma** *mustT-If*:  
**assumes** *mustT (If tst c1 c2) s*  
**shows** *mustT c1 s*  $\vee$  *mustT c2 s*  
 $\langle$ *proof* $\rangle$

**lemma** *mustT-Par-L*:  
**assumes** *mustT (Par c1 c2) s*  
**shows** *mustT c1 s*  
 $\langle$ *proof* $\rangle$

**lemma** *mustT-Par-R*:  
**assumes** *mustT (Par c1 c2) s*  
**shows** *mustT c2 s*  
 $\langle$ *proof* $\rangle$

**definition** *determOn* **where**  
*determOn phi r*  $\equiv$   
 $\forall a b b'. \text{phi } a \wedge r a b \wedge r a b' \longrightarrow b = b'$

**lemma** *determOn-seq-transT*:  
*determOn*  $(\lambda(c,s). \text{seq } c)$  *transT*  
 $\langle$ *proof* $\rangle$

**end**

**end**

## 4 During-execution security

**theory** *During-Execution*  
**imports** *Bisim Language-Semantics* **begin**

### 4.1 Basic setting

**locale** *PL-Indis* = *PL tval aval*  
**for**



$tval :: 'test \Rightarrow 'state \Rightarrow bool$  **and**  
 $aval :: 'atom \Rightarrow 'state \Rightarrow 'state$   
 +  
**fixes**  
 $indis :: 'state \text{ rel}$   
**assumes**  
 $equiv-indis: equiv \ UNIV \ indis$

**context**  $PL-Indis$   
**begin**

**abbreviation**  $indisAbbrev$  (**infix**  $\approx 50$ )  
**where**  $s1 \approx s2 \equiv (s1, s2) \in indis$

**definition**  $indisE$  (**infix**  $\approx_e 50$ ) **where**  
 $se1 \approx_e se2 \equiv$   
*case*  $(se1, se2)$  *of*  
 $(Inl \ s1, Inl \ s2) \Rightarrow s1 \approx s2$   
 $|(Inr \ err1, Inr \ err2) \Rightarrow err1 = err2$

**lemma**  $refl-indis: refl \ indis$   
**and**  $trans-indis: trans \ indis$   
**and**  $sym-indis: sym \ indis$   
 $\langle proof \rangle$

**lemma**  $indis-refl[intro]: s \approx s$   
 $\langle proof \rangle$

**lemma**  $indis-trans: \llbracket s \approx s'; s' \approx s'' \rrbracket \Longrightarrow s \approx s''$   
 $\langle proof \rangle$

**lemma**  $indis-sym: s \approx s' \Longrightarrow s' \approx s$   
 $\langle proof \rangle$

## 4.2 Compatibility and discreteness

**definition**  $compatTst$  **where**  
 $compatTst \ tst \equiv$   
 $\forall \ s \ t. s \approx t \longrightarrow tval \ tst \ s = tval \ tst \ t$

**definition**  $compatAtm$  **where**  
 $compatAtm \ atm \equiv$   
 $\forall \ s \ t. s \approx t \longrightarrow aval \ atm \ s \approx aval \ atm \ t$

**definition**  $presAtm$  **where**  
 $presAtm \ atm \equiv$

$\forall s. s \approx \text{aval atm } s$

**coinductive** *discr where*

*intro:*

$\llbracket \bigwedge s c' s'. (c,s) \rightarrow c (c',s') \implies s \approx s' \wedge \text{discr } c';$   
 $\bigwedge s s'. (c,s) \rightarrow t s' \implies s \approx s' \rrbracket$   
 $\implies \text{discr } c$

**lemma** *presAtm-compatAtm[simp]:*

**assumes** *presAtm atm*

**shows** *compatAtm atm*

*\langle proof \rangle*

Coinduction for discreteness:

**lemma** *discr-coind:*

**assumes** \*: *phi c and*

\*\* :  $\bigwedge c s c' s'. \llbracket \text{phi } c; (c,s) \rightarrow c (c',s') \rrbracket \implies s \approx s' \wedge (\text{phi } c' \vee \text{discr } c')$  **and**

\*\*\* :  $\bigwedge c s s'. \llbracket \text{phi } c; (c,s) \rightarrow t s' \rrbracket \implies s \approx s'$

**shows** *discr c*

*\langle proof \rangle*

**lemma** *discr-raw-coind:*

**assumes** \*: *phi c and*

\*\* :  $\bigwedge c s c' s'. \llbracket \text{phi } c; (c,s) \rightarrow c (c',s') \rrbracket \implies s \approx s' \wedge \text{phi } c'$  **and**

\*\*\* :  $\bigwedge c s s'. \llbracket \text{phi } c; (c,s) \rightarrow t s' \rrbracket \implies s \approx s'$

**shows** *discr c*

*\langle proof \rangle*

Discreteness versus transition:

**lemma** *discr-transC:*

**assumes** \*: *discr c and* \*\*:  $(c,s) \rightarrow c (c',s')$

**shows** *discr c'*

*\langle proof \rangle*

**lemma** *discr-MtransC:*

**assumes** *discr c and*  $(c,s) \rightarrow *c (c',s')$

**shows** *discr c'*

*\langle proof \rangle*

**lemma** *discr-transC-indis:*

**assumes** \*: *discr c and* \*\*:  $(c,s) \rightarrow c (c',s')$

**shows**  $s \approx s'$

*\langle proof \rangle*

**lemma** *discr-MtransC-indis:*

**assumes** *discr c and*  $(c,s) \rightarrow *c (c',s')$

**shows**  $s \approx s'$

*\langle proof \rangle*

**lemma** *discr-transT*:  
**assumes** \*: *discr c* **and** \*\*:  $(c,s) \rightarrow t s'$   
**shows**  $s \approx s'$   
 $\langle proof \rangle$

**lemma** *discr-MtransT*:  
**assumes** \*: *discr c* **and** \*\*:  $(c,s) \rightarrow *t s'$   
**shows**  $s \approx s'$   
 $\langle proof \rangle$

### 4.3 Terminating-interactive discreteness

**coinductive** *discr0* **where**

*intro*:

$\llbracket \bigwedge s c' s'. \llbracket mustT c s; (c,s) \rightarrow c (c',s') \rrbracket \implies s \approx s' \wedge discr0 c' ;$   
 $\bigwedge s s'. \llbracket mustT c s; (c,s) \rightarrow t s \rrbracket \implies s \approx s \rrbracket$   
 $\implies discr0 c$

Coinduction for 0-discreteness:

**lemma** *discr0-coind*[*consumes 1, case-names Cont Term, induct pred: discr0*]:

**assumes** \*: *phi c* **and**

\*\* :  $\bigwedge c s c' s'$ .

$\llbracket mustT c s; phi c; (c,s) \rightarrow c (c',s') \rrbracket \implies$   
 $s \approx s' \wedge (phi c' \vee discr0 c') \mathbf{and}$

\*\*\* :  $\bigwedge c s s'. \llbracket mustT c s; phi c; (c,s) \rightarrow t s \rrbracket \implies s \approx s'$

**shows** *discr0 c*

$\langle proof \rangle$

**lemma** *discr0-raw-coind*[*consumes 1, case-names Cont Term*]:

**assumes** \*: *phi c* **and**

\*\* :  $\bigwedge c s c' s'. \llbracket mustT c s; phi c; (c,s) \rightarrow c (c',s') \rrbracket \implies s \approx s' \wedge phi c' \mathbf{and}$

\*\*\* :  $\bigwedge c s s'. \llbracket mustT c s; phi c; (c,s) \rightarrow t s \rrbracket \implies s \approx s'$

**shows** *discr0 c*

$\langle proof \rangle$

0-Discreteness versus transition:

**lemma** *discr0-transC*:

**assumes** \*: *discr0 c* **and** \*\*:  $mustT c s (c,s) \rightarrow c (c',s')$

**shows** *discr0 c'*

$\langle proof \rangle$

**lemma** *discr0-MtransC*:

**assumes** *discr0 c* **and**  $mustT c s (c,s) \rightarrow *c (c',s')$

**shows** *discr0 c'*

$\langle proof \rangle$

**lemma** *discr0-transC-indis*:

**assumes** \*: *discr0 c* **and** \*\*:  $mustT c s (c,s) \rightarrow c (c',s')$

**shows**  $s \approx s'$

$\langle \text{proof} \rangle$

**lemma** *discr0-MtransC-indis*:

**assumes** *discr0 c* **and** *mustT c s (c,s)  $\rightarrow^* c (c',s')$*

**shows**  $s \approx s'$

$\langle \text{proof} \rangle$

**lemma** *discr0-transT*:

**assumes**  $*$ : *discr0 c* **and**  $**$ : *mustT c s (c,s)  $\rightarrow t s'$*

**shows**  $s \approx s'$

$\langle \text{proof} \rangle$

**lemma** *discr0-MtransT*:

**assumes**  $*$ : *discr0 c* **and**  $***$ : *mustT c s* **and**  $**$ :  $(c,s) \rightarrow^* t s'$

**shows**  $s \approx s'$

$\langle \text{proof} \rangle$

**lemma** *discr-discr0[simp]*: *discr c  $\implies$  discr0 c*

$\langle \text{proof} \rangle$

## 4.4 Self-isomorphism

**coinductive** *siso* **where**

*intro*:

$\llbracket \bigwedge s c' s'. (c,s) \rightarrow c (c',s') \implies \text{siso } c' \rrbracket$ ;

$\bigwedge s t c' s'. \llbracket s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$ ;

$\bigwedge s t s'. \llbracket s \approx t; (c,s) \rightarrow t s' \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$

$\implies \text{siso } c$

Coinduction for self-isomorphism:

**lemma** *siso-coind*:

**assumes**  $*$ : *phi c* **and**

$**$ :  $\bigwedge c s c' s'. \llbracket \text{phi } c; (c,s) \rightarrow c (c',s') \rrbracket \implies \text{phi } c' \vee \text{siso } c'$  **and**

$***$ :  $\bigwedge c s t c' s'. \llbracket \text{phi } c; s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$

**and**

$****$ :  $\bigwedge c s t s'. \llbracket \text{phi } c; s \approx t; (c,s) \rightarrow t s' \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$

**shows** *siso c*

$\langle \text{proof} \rangle$

**lemma** *siso-raw-coind*:

**assumes**  $*$ : *phi c* **and**

$**$ :  $\bigwedge c s c' s'. \llbracket \text{phi } c; (c,s) \rightarrow c (c',s') \rrbracket \implies \text{phi } c'$  **and**

$***$ :  $\bigwedge c s t c' s'. \llbracket \text{phi } c; s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$

**and**

$****$ :  $\bigwedge c s t s'. \llbracket \text{phi } c; s \approx t; (c,s) \rightarrow t s' \rrbracket \implies \exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$

**shows** *siso c*

$\langle \text{proof} \rangle$

Self-Isomorphism versus transition:

**lemma** *siso-transC*:  
**assumes** \*: *siso c* **and** \*\*:  $(c,s) \rightarrow c (c',s')$   
**shows** *siso c'*  
 $\langle proof \rangle$

**lemma** *siso-MtransC*:  
**assumes** *siso c* **and**  $(c,s) \rightarrow *c (c',s')$   
**shows** *siso c'*  
 $\langle proof \rangle$

**lemma** *siso-transC-indis*:  
**assumes** \*: *siso c* **and** \*\*:  $(c,s) \rightarrow c (c',s')$  **and** \*\*\*:  $s \approx t$   
**shows**  $\exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$   
 $\langle proof \rangle$

**lemma** *siso-transT*:  
**assumes** \*: *siso c* **and** \*\*:  $(c,s) \rightarrow t s'$  **and** \*\*\*:  $s \approx t$   
**shows**  $\exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$   
 $\langle proof \rangle$

## 4.5 MustT-interactive self-isomorphism

**coinductive** *siso0* **where**

*intro*:

$\llbracket \bigwedge s c' s'. \llbracket mustT c s; (c,s) \rightarrow c (c',s') \rrbracket \implies siso0 c';$   
 $\bigwedge s t c' s'.$   
 $\llbracket mustT c s; mustT c t; s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies$   
 $\exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t');$   
 $\bigwedge s t s'.$   
 $\llbracket mustT c s; mustT c t; s \approx t; (c,s) \rightarrow t s' \rrbracket \implies$   
 $\exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$   
 $\implies siso0 c$

Coinduction for self-isomorphism:

**lemma** *siso0-coind*[*consumes 1, case-names Indef Cont Term, induct pred: discr0*]:

**assumes** \*: *phi c* **and**

\*\* :  $\bigwedge c s c' s'. \llbracket phi c; mustT c s; (c,s) \rightarrow c (c',s') \rrbracket \implies phi c' \vee siso0 c'$  **and**

\*\*\* :  $\bigwedge c s t c' s'.$

$\llbracket phi c; mustT c s; mustT c t; s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies$

$\exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$  **and**

\*\*\*\* :  $\bigwedge c s t s'.$

$\llbracket mustT c s; mustT c t; phi c; s \approx t; (c,s) \rightarrow t s' \rrbracket \implies$

$\exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$

**shows** *siso0 c*

$\langle proof \rangle$

**lemma** *siso0-raw-coind*[*consumes 1, case-names Indef Cont Term*]:

**assumes** \*: *phi c* **and**

\*\* :  $\bigwedge c s c' s'. \llbracket phi c; mustT c s; (c,s) \rightarrow c (c',s') \rrbracket \implies phi c'$  **and**

**\*\*\*:**  $\bigwedge c s t c' s'$ .  
 $\llbracket \text{phi } c; \text{ mustT } c s; \text{ mustT } c t; s \approx t; (c,s) \rightarrow c (c',s') \rrbracket \implies$   
 $\exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$  **and**  
**\*\*\*\*:**  $\bigwedge c s t s'$ .  
 $\llbracket \text{phi } c; \text{ mustT } c s; \text{ mustT } c t; s \approx t; (c,s) \rightarrow t s' \rrbracket \implies$   
 $\exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$   
**shows** *siso0 c*  
 $\langle \text{proof} \rangle$

Self-Isomorphism versus transition:

**lemma** *siso0-transC*:  
**assumes** \*: *siso0 c* **and** \*\*: *mustT c s (c,s)  $\rightarrow c (c',s')$*   
**shows** *siso0 c'*  
 $\langle \text{proof} \rangle$

**lemma** *siso0-MtransC*:  
**assumes** *siso0 c* **and** *mustT c s* **and**  $(c,s) \rightarrow *c (c',s')$   
**shows** *siso0 c'*  
 $\langle \text{proof} \rangle$

**lemma** *siso0-transC-indis*:  
**assumes** \*: *siso0 c*  
**and** \*\*: *mustT c s mustT c t (c,s)  $\rightarrow c (c',s')$*   
**and** \*\*\*:  $s \approx t$   
**shows**  $\exists t'. s' \approx t' \wedge (c,t) \rightarrow c (c',t')$   
 $\langle \text{proof} \rangle$

**lemma** *siso0-transT*:  
**assumes** \*: *siso0 c*  
**and** \*\*: *mustT c s mustT c t (c,s)  $\rightarrow t s'$*   
**and** \*\*\*:  $s \approx t$   
**shows**  $\exists t'. s' \approx t' \wedge (c,t) \rightarrow t t'$   
 $\langle \text{proof} \rangle$

## 4.6 Notions of bisimilarity

Matchers:

**definition** *matchC-C* **where**  
*matchC-C* *theta c d*  $\equiv$   
 $\forall s t c' s'$ .  
 $s \approx t \wedge (c,s) \rightarrow c (c',s')$   
 $\longrightarrow$   
 $(\exists d' t'. (d,t) \rightarrow c (d',t') \wedge s' \approx t' \wedge (c',d') \in \text{theta})$

**definition** *matchC-ZOC* **where**  
*matchC-ZOC* *theta c d*  $\equiv$   
 $\forall s t c' s'$ .  
 $s \approx t \wedge (c,s) \rightarrow c (c',s')$   
 $\longrightarrow$

$$\begin{aligned}
& (s' \approx t \wedge (c', d) \in \text{theta}) \\
& \vee \\
& (\exists d' t'. (d, t) \rightarrow c (d', t') \wedge s' \approx t' \wedge (c', d') \in \text{theta})
\end{aligned}$$

**definition** *matchC-ZO* where

$$\text{matchC-ZO } \text{theta } c \ d \equiv$$

$$\begin{aligned}
& \forall s \ t \ c' \ s'. \\
& \quad s \approx t \wedge (c, s) \rightarrow c (c', s') \\
& \quad \longrightarrow \\
& \quad (s' \approx t \wedge (c', d) \in \text{theta}) \\
& \quad \vee \\
& \quad (\exists d' t'. (d, t) \rightarrow c (d', t') \wedge s' \approx t' \wedge (c', d') \in \text{theta}) \\
& \quad \vee \\
& \quad (\exists t'. (d, t) \rightarrow t \ t' \wedge s' \approx t' \wedge \text{discr } c')
\end{aligned}$$

**definition** *matchT-T* where

$$\text{matchT-T } c \ d \equiv$$

$$\begin{aligned}
& \forall s \ t \ s'. \\
& \quad s \approx t \wedge (c, s) \rightarrow t \ s' \\
& \quad \longrightarrow \\
& \quad (\exists t'. (d, t) \rightarrow t \ t' \wedge s' \approx t')
\end{aligned}$$

**definition** *matchT-ZO* where

$$\text{matchT-ZO } c \ d \equiv$$

$$\begin{aligned}
& \forall s \ t \ s'. \\
& \quad s \approx t \wedge (c, s) \rightarrow t \ s' \\
& \quad \longrightarrow \\
& \quad (s' \approx t \wedge \text{discr } d) \\
& \quad \vee \\
& \quad (\exists d' t'. (d, t) \rightarrow c (d', t') \wedge s' \approx t' \wedge \text{discr } d') \\
& \quad \vee \\
& \quad (\exists t'. (d, t) \rightarrow t \ t' \wedge s' \approx t')
\end{aligned}$$

**definition** *matchC-MC* where

$$\text{matchC-MC } \text{theta } c \ d \equiv$$

$$\begin{aligned}
& \forall s \ t \ c' \ s'. \\
& \quad s \approx t \wedge (c, s) \rightarrow c (c', s') \\
& \quad \longrightarrow \\
& \quad (\exists d' t'. (d, t) \rightarrow *c (d', t') \wedge s' \approx t' \wedge (c', d') \in \text{theta})
\end{aligned}$$

**definition** *matchC-TMC* where

$$\text{matchC-TMC } \text{theta } c \ d \equiv$$

$$\begin{aligned}
& \forall s \ t \ c' \ s'. \\
& \quad \text{mustT } c \ s \wedge \text{mustT } d \ t \wedge s \approx t \wedge (c, s) \rightarrow c (c', s') \\
& \quad \longrightarrow \\
& \quad (\exists d' t'. (d, t) \rightarrow *c (d', t') \wedge s' \approx t' \wedge (c', d') \in \text{theta})
\end{aligned}$$

**definition** *matchC-M* where

*matchC-M*  $\theta$   $c$   $d \equiv$

$\forall s t c' s'.$

$s \approx t \wedge (c, s) \rightarrow c (c', s')$

$\rightarrow$

$(\exists d' t'. (d, t) \rightarrow * c (d', t') \wedge s' \approx t' \wedge (c', d') \in \theta)$

$\vee$

$(\exists t'. (d, t) \rightarrow * t t' \wedge s' \approx t' \wedge \text{discr } c')$

**definition** *matchT-MT* where

*matchT-MT*  $c$   $d \equiv$

$\forall s t s'.$

$s \approx t \wedge (c, s) \rightarrow t s'$

$\rightarrow$

$(\exists t'. (d, t) \rightarrow * t t' \wedge s' \approx t')$

**definition** *matchT-TMT* where

*matchT-TMT*  $c$   $d \equiv$

$\forall s t s'.$

$\text{mustT } c s \wedge \text{mustT } d t \wedge s \approx t \wedge (c, s) \rightarrow t s'$

$\rightarrow$

$(\exists t'. (d, t) \rightarrow * t t' \wedge s' \approx t')$

**definition** *matchT-M* where

*matchT-M*  $c$   $d \equiv$

$\forall s t s'.$

$s \approx t \wedge (c, s) \rightarrow t s'$

$\rightarrow$

$(\exists d' t'. (d, t) \rightarrow * c (d', t') \wedge s' \approx t' \wedge \text{discr } d')$

$\vee$

$(\exists t'. (d, t) \rightarrow * t t' \wedge s' \approx t')$

**lemmas** *match-defs* =

*matchC-C-def*

*matchC-ZOC-def* *matchC-ZO-def*

*matchT-T-def* *matchT-ZO-def*

*matchC-MC-def* *matchC-M-def*

*matchT-MT-def* *matchT-M-def*

*matchC-TMC-def* *matchT-TMT-def*

**lemma** *matchC-C-def2*:

*matchC-C*  $\theta$   $d$   $c =$

$(\forall s t d' t'.$

$s \approx t \wedge (d, t) \rightarrow c (d', t')$

$\rightarrow$

$(\exists c' s'. (c, s) \rightarrow c (c', s') \wedge s' \approx t' \wedge (d', c') \in \theta))$

*<proof>*



**lemma** *matchC-ZOC-def2*:  
*matchC-ZOC* *theta* *d* *c* =  
 $(\forall s t d' t'.$   
 $s \approx t \wedge (d, t) \rightarrow c (d', t')$   
 $\rightarrow$   
 $(s \approx t' \wedge (d', c) \in \text{theta})$   
 $\vee$   
 $(\exists c' s'. (c, s) \rightarrow c (c', s') \wedge s' \approx t' \wedge (d', c') \in \text{theta}))$   
 $\langle \text{proof} \rangle$

**lemma** *matchC-ZO-def2*:  
*matchC-ZO* *theta* *d* *c* =  
 $(\forall s t d' t'.$   
 $s \approx t \wedge (d, t) \rightarrow c (d', t')$   
 $\rightarrow$   
 $(s \approx t' \wedge (d', c) \in \text{theta})$   
 $\vee$   
 $(\exists c' s'. (c, s) \rightarrow c (c', s') \wedge s' \approx t' \wedge (d', c') \in \text{theta})$   
 $\vee$   
 $(\exists s'. (c, s) \rightarrow t s' \wedge s' \approx t' \wedge \text{discr } d')$   
 $\langle \text{proof} \rangle$

**lemma** *matchT-T-def2*:  
*matchT-T* *d* *c* =  
 $(\forall s t t'.$   
 $s \approx t \wedge (d, t) \rightarrow t t'$   
 $\rightarrow$   
 $(\exists s'. (c, s) \rightarrow t s' \wedge s' \approx t')$   
 $\langle \text{proof} \rangle$

**lemma** *matchT-ZO-def2*:  
*matchT-ZO* *d* *c* =  
 $(\forall s t t'.$   
 $s \approx t \wedge (d, t) \rightarrow t t'$   
 $\rightarrow$   
 $(s \approx t' \wedge \text{discr } c)$   
 $\vee$   
 $(\exists c' s'. (c, s) \rightarrow c (c', s') \wedge s' \approx t' \wedge \text{discr } c')$   
 $\vee$   
 $(\exists s'. (c, s) \rightarrow t s' \wedge s' \approx t')$   
 $\langle \text{proof} \rangle$

**lemma** *matchC-MC-def2*:  
*matchC-MC* *theta* *d* *c* =  
 $(\forall s t d' t'.$   
 $s \approx t \wedge (d, t) \rightarrow c (d', t')$

$\longrightarrow$   
 $(\exists c' s'. (c,s) \rightarrow *c (c',s') \wedge s' \approx t' \wedge (d',c') \in \text{theta}))$   
 $\langle \text{proof} \rangle$

**lemma** *matchC-TMC-def2*:  
 $\text{matchC-TMC } \text{theta } d \text{ } c =$   
 $(\forall s t d' t'.$   
 $\text{mustT } c \ s \wedge \text{mustT } d \ t \wedge s \approx t \wedge (d,t) \rightarrow c (d',t')$   
 $\longrightarrow$   
 $(\exists c' s'. (c,s) \rightarrow *c (c',s') \wedge s' \approx t' \wedge (d',c') \in \text{theta}))$   
 $\langle \text{proof} \rangle$

**lemma** *matchC-M-def2*:  
 $\text{matchC-M } \text{theta } d \ c =$   
 $(\forall s t d' t'.$   
 $s \approx t \wedge (d,t) \rightarrow c (d',t')$   
 $\longrightarrow$   
 $(\exists c' s'. (c,s) \rightarrow *c (c',s') \wedge s' \approx t' \wedge (d',c') \in \text{theta})$   
 $\vee$   
 $(\exists s'. (c,s) \rightarrow *t s' \wedge s' \approx t' \wedge \text{discr } d'))$   
 $\langle \text{proof} \rangle$

**lemma** *matchT-MT-def2*:  
 $\text{matchT-MT } d \ c =$   
 $(\forall s t t'.$   
 $s \approx t \wedge (d,t) \rightarrow t t'$   
 $\longrightarrow$   
 $(\exists s'. (c,s) \rightarrow *t s' \wedge s' \approx t'))$   
 $\langle \text{proof} \rangle$

**lemma** *matchT-TMT-def2*:  
 $\text{matchT-TMT } d \ c =$   
 $(\forall s t t'.$   
 $\text{mustT } c \ s \wedge \text{mustT } d \ t \wedge s \approx t \wedge (d,t) \rightarrow t t'$   
 $\longrightarrow$   
 $(\exists s'. (c,s) \rightarrow *t s' \wedge s' \approx t'))$   
 $\langle \text{proof} \rangle$

**lemma** *matchT-M-def2*:  
 $\text{matchT-M } d \ c =$   
 $(\forall s t t'.$   
 $s \approx t \wedge (d,t) \rightarrow t t'$   
 $\longrightarrow$   
 $(\exists c' s'. (c,s) \rightarrow *c (c',s') \wedge s' \approx t' \wedge \text{discr } c')$   
 $\vee$   
 $(\exists s'. (c,s) \rightarrow *t s' \wedge s' \approx t'))$   
 $\langle \text{proof} \rangle$

Retracts:

**definition** *Sretr* **where**

$Sretr\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-C\ theta\ c\ d \wedge$   
 $\quad matchT-T\ c\ d\}$

**definition** *ZOretr* **where**

$ZOretr\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-ZO\ theta\ c\ d \wedge$   
 $\quad matchT-ZO\ c\ d\}$

**definition** *ZOretrT* **where**

$ZOretrT\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-ZOC\ theta\ c\ d \wedge$   
 $\quad matchT-T\ c\ d\}$

**definition** *Wretr* **where**

$Wretr\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-M\ theta\ c\ d \wedge$   
 $\quad matchT-M\ c\ d\}$

**definition** *WretrT* **where**

$WretrT\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-MC\ theta\ c\ d \wedge$   
 $\quad matchT-MT\ c\ d\}$

**definition** *RetrT* **where**

$RetrT\ theta \equiv$   
 $\{(c,d).$   
 $\quad matchC-TMC\ theta\ c\ d \wedge$   
 $\quad matchT-TMT\ c\ d\}$

**lemmas** *Retr-defs* =

*Sretr-def*  
*ZOretr-def ZOretrT-def*  
*Wretr-def WretrT-def*  
*RetrT-def*

The associated bisimilarity relations:

**definition** *Sbis* **where**  $Sbis \equiv bis\ Sretr$

**definition**  $ZObis$  **where**  $ZObis \equiv bis\ ZOretr$   
**definition**  $ZObisT$  **where**  $ZObisT \equiv bis\ ZOretrT$   
**definition**  $Wbis$  **where**  $Wbis \equiv bis\ Wretr$   
**definition**  $WbisT$  **where**  $WbisT \equiv bis\ WretrT$   
**definition**  $BisT$  **where**  $BisT \equiv bis\ RetrT$

**lemmas**  $bis-defs =$   
 $Sbis-def$   
 $ZObis-def\ ZObisT-def$   
 $Wbis-def\ WbisT-def$   
 $BisT-def$

**abbreviation**  $Sbis-abbrev$  (**infix**  $\approx_s$  55) **where**  $c1 \approx_s c2 \equiv (c1, c2) \in Sbis$   
**abbreviation**  $ZObis-abbrev$  (**infix**  $\approx_{01}$  55) **where**  $c1 \approx_{01} c2 \equiv (c1, c2) \in ZObis$   
**abbreviation**  $ZObisT-abbrev$  (**infix**  $\approx_{01T}$  55) **where**  $c1 \approx_{01T} c2 \equiv (c1, c2) \in ZObisT$   
**abbreviation**  $Wbis-abbrev$  (**infix**  $\approx_w$  55) **where**  $c1 \approx_w c2 \equiv (c1, c2) \in Wbis$   
**abbreviation**  $WbisT-abbrev$  (**infix**  $\approx_{wT}$  55) **where**  $c1 \approx_{wT} c2 \equiv (c1, c2) \in WbisT$   
**abbreviation**  $BisT-abbrev$  (**infix**  $\approx_T$  55) **where**  $c1 \approx_T c2 \equiv (c1, c2) \in BisT$

**lemma**  $mono-Retr$ :  
 $mono\ Sretr$   
 $mono\ ZOretr\ mono\ ZOretrT$   
 $mono\ Wretr\ mono\ WretrT$   
 $mono\ RetrT$   
 $\langle proof \rangle$

**lemma**  $Sbis-prefix$ :  
 $Sbis \subseteq Sretr\ Sbis$   
 $\langle proof \rangle$

**lemma**  $Sbis-sym$ :  $sym\ Sbis$   
 $\langle proof \rangle$

**lemma**  $Sbis-Sym$ :  $c \approx_s d \implies d \approx_s c$   
 $\langle proof \rangle$

**lemma**  $Sbis-converse$ :  
 $((c, d) \in theta^{-1} \cup Sbis) = ((d, c) \in theta \cup Sbis)$   
 $\langle proof \rangle$

**lemma**  
 $Sbis-matchC-C$ :  $\bigwedge s\ t. c \approx_s d \implies matchC-C\ Sbis\ c\ d$   
**and**  
 $Sbis-matchT-T$ :  $\bigwedge c\ d. c \approx_s d \implies matchT-T\ c\ d$   
 $\langle proof \rangle$

**lemmas**  $Sbis\text{-step} = Sbis\text{-matchC-C } Sbis\text{-matchT-T}$

**lemma**

$Sbis\text{-matchC-C-rev}: \bigwedge s t. s \approx s t \implies matchC-C Sbis t s$

**and**

$Sbis\text{-matchT-T-rev}: \bigwedge s t. s \approx s t \implies matchT-T t s$

$\langle proof \rangle$

**lemmas**  $Sbis\text{-step-rev} = Sbis\text{-matchC-C-rev } Sbis\text{-matchT-T-rev}$

**lemma**  $Sbis\text{-coind}$ :

**assumes**  $sym\ theta$  **and**  $theta \subseteq Sretr (theta \cup Sbis)$

**shows**  $theta \subseteq Sbis$

$\langle proof \rangle$

**lemma**  $Sbis\text{-raw-coind}$ :

**assumes**  $sym\ theta$  **and**  $theta \subseteq Sretr\ theta$

**shows**  $theta \subseteq Sbis$

$\langle proof \rangle$

**lemma**  $Sbis\text{-coind2}$ :

**assumes**  $theta \subseteq Sretr (theta \cup Sbis)$  **and**

$theta \hat{-} 1 \subseteq Sretr ((theta \hat{-} 1) \cup Sbis)$

**shows**  $theta \subseteq Sbis$

$\langle proof \rangle$

**lemma**  $Sbis\text{-raw-coind2}$ :

**assumes**  $theta \subseteq Sretr\ theta$  **and**

$theta \hat{-} 1 \subseteq Sretr (theta \hat{-} 1)$

**shows**  $theta \subseteq Sbis$

$\langle proof \rangle$

**lemma**  $ZObis\text{-prefix}$ :

$ZObis \subseteq ZOretr\ ZObis$

$\langle proof \rangle$

**lemma**  $ZObis\text{-sym}$ :  $sym\ ZObis$

$\langle proof \rangle$

**lemma**  $ZObis\text{-converse}$ :

$((c,d) \in theta \hat{-} 1 \cup ZObis) = ((d,c) \in theta \cup ZObis)$

$\langle proof \rangle$

**lemma**  $ZObis\text{-Sym}$ :  $s \approx 01 t \implies t \approx 01 s$

$\langle proof \rangle$

**lemma**

*ZObis-matchC-ZO*:  $\bigwedge s t. s \approx 01 t \implies \text{matchC-ZO } ZObis s t$

**and**

*ZObis-matchT-ZO*:  $\bigwedge s t. s \approx 01 t \implies \text{matchT-ZO } s t$

*<proof>*

**lemmas** *ZObis-step* = *ZObis-matchC-ZO* *ZObis-matchT-ZO*

**lemma**

*ZObis-matchC-ZO-rev*:  $\bigwedge s t. s \approx 01 t \implies \text{matchC-ZO } ZObis t s$

**and**

*ZObis-matchT-ZO-rev*:  $\bigwedge s t. s \approx 01 t \implies \text{matchT-ZO } t s$

*<proof>*

**lemmas** *ZObis-step-rev* = *ZObis-matchC-ZO-rev* *ZObis-matchT-ZO-rev*

**lemma** *ZObis-coind*:

**assumes** *sym theta* **and**  $theta \subseteq ZOrtr (theta \cup ZObis)$

**shows**  $theta \subseteq ZObis$

*<proof>*

**lemma** *ZObis-raw-coind*:

**assumes** *sym theta* **and**  $theta \subseteq ZOrtr theta$

**shows**  $theta \subseteq ZObis$

*<proof>*

**lemma** *ZObis-coind2*:

**assumes**  $theta \subseteq ZOrtr (theta \cup ZObis)$  **and**

$theta \hat{-} 1 \subseteq ZOrtr ((theta \hat{-} 1) \cup ZObis)$

**shows**  $theta \subseteq ZObis$

*<proof>*

**lemma** *ZObis-raw-coind2*:

**assumes**  $theta \subseteq ZOrtr theta$  **and**

$theta \hat{-} 1 \subseteq ZOrtr (theta \hat{-} 1)$

**shows**  $theta \subseteq ZObis$

*<proof>*

**lemma** *ZObisT-prefix*:

$ZObisT \subseteq ZOrtrT ZObisT$

*<proof>*

**lemma** *ZObisT-sym*: *sym ZObisT*

*<proof>*

**lemma** *ZObisT-Sym*:  $s \approx 01T t \implies t \approx 01T s$

*<proof>*

**lemma** *ZObisT-converse*:

$((c,d) \in \text{theta}^{\wedge-1} \cup \text{ZObisT}) = ((d,c) \in \text{theta} \cup \text{ZObisT})$   
 $\langle \text{proof} \rangle$

**lemma**

$\text{ZObisT-matchC-ZOC}: \bigwedge s t. s \approx 01T t \implies \text{matchC-ZOC ZObisT } s t$

**and**

$\text{ZObisT-matchT-T}: \bigwedge s t. s \approx 01T t \implies \text{matchT-T } s t$

$\langle \text{proof} \rangle$

**lemmas**  $\text{ZObisT-step} = \text{ZObisT-matchC-ZOC ZObisT-matchT-T}$

**lemma**

$\text{ZObisT-matchC-ZOC-rev}: \bigwedge s t. s \approx 01T t \implies \text{matchC-ZOC ZObisT } t s$

**and**

$\text{ZObisT-matchT-T-rev}: \bigwedge s t. s \approx 01T t \implies \text{matchT-T } t s$

$\langle \text{proof} \rangle$

**lemmas**  $\text{ZObisT-step-rev} = \text{ZObisT-matchC-ZOC-rev ZObisT-matchT-T-rev}$

**lemma**  $\text{ZObisT-coind}$ :

**assumes**  $\text{sym } \text{theta}$  **and**  $\text{theta} \subseteq \text{ZOretrT } (\text{theta} \cup \text{ZObisT})$

**shows**  $\text{theta} \subseteq \text{ZObisT}$

$\langle \text{proof} \rangle$

**lemma**  $\text{ZObisT-raw-coind}$ :

**assumes**  $\text{sym } \text{theta}$  **and**  $\text{theta} \subseteq \text{ZOretrT } \text{theta}$

**shows**  $\text{theta} \subseteq \text{ZObisT}$

$\langle \text{proof} \rangle$

**lemma**  $\text{ZObisT-coind2}$ :

**assumes**  $\text{theta} \subseteq \text{ZOretrT } (\text{theta} \cup \text{ZObisT})$  **and**

$\text{theta}^{\wedge-1} \subseteq \text{ZOretrT } ((\text{theta}^{\wedge-1}) \cup \text{ZObisT})$

**shows**  $\text{theta} \subseteq \text{ZObisT}$

$\langle \text{proof} \rangle$

**lemma**  $\text{ZObisT-raw-coind2}$ :

**assumes**  $\text{theta} \subseteq \text{ZOretrT } \text{theta}$  **and**

$\text{theta}^{\wedge-1} \subseteq \text{ZOretrT } (\text{theta}^{\wedge-1})$

**shows**  $\text{theta} \subseteq \text{ZObisT}$

$\langle \text{proof} \rangle$

**lemma**  $\text{Wbis-prefix}$ :

$\text{Wbis} \subseteq \text{Wretr } \text{Wbis}$

$\langle \text{proof} \rangle$

**lemma**  $\text{Wbis-sym}$ :  $\text{sym } \text{Wbis}$

$\langle \text{proof} \rangle$

**lemma** *Wbis-converse*:  
 $((c,d) \in \text{theta}^{-1} \cup \text{Wbis}) = ((d,c) \in \text{theta} \cup \text{Wbis})$   
 $\langle \text{proof} \rangle$

**lemma** *Wbis-Sym*:  $c \approx_w d \implies d \approx_w c$   
 $\langle \text{proof} \rangle$

**lemma**  
*Wbis-matchC-M*:  $\bigwedge c d. c \approx_w d \implies \text{matchC-M } \text{Wbis } c d$   
**and**  
*Wbis-matchT-M*:  $\bigwedge c d. c \approx_w d \implies \text{matchT-M } c d$   
 $\langle \text{proof} \rangle$

**lemmas**  $\text{Wbis-step} = \text{Wbis-matchC-M } \text{Wbis-matchT-M}$

**lemma**  
*Wbis-matchC-M-rev*:  $\bigwedge s t. s \approx_w t \implies \text{matchC-M } \text{Wbis } t s$   
**and**  
*Wbis-matchT-M-rev*:  $\bigwedge s t. s \approx_w t \implies \text{matchT-M } t s$   
 $\langle \text{proof} \rangle$

**lemmas**  $\text{Wbis-step-rev} = \text{Wbis-matchC-M-rev } \text{Wbis-matchT-M-rev}$

**lemma** *Wbis-coind*:  
**assumes**  $\text{sym } \text{theta}$  **and**  $\text{theta} \subseteq \text{Wretr } (\text{theta} \cup \text{Wbis})$   
**shows**  $\text{theta} \subseteq \text{Wbis}$   
 $\langle \text{proof} \rangle$

**lemma** *Wbis-raw-coind*:  
**assumes**  $\text{sym } \text{theta}$  **and**  $\text{theta} \subseteq \text{Wretr } \text{theta}$   
**shows**  $\text{theta} \subseteq \text{Wbis}$   
 $\langle \text{proof} \rangle$

**lemma** *Wbis-coind2*:  
**assumes**  $\text{theta} \subseteq \text{Wretr } (\text{theta} \cup \text{Wbis})$  **and**  
 $\text{theta}^{-1} \subseteq \text{Wretr } ((\text{theta}^{-1}) \cup \text{Wbis})$   
**shows**  $\text{theta} \subseteq \text{Wbis}$   
 $\langle \text{proof} \rangle$

**lemma** *Wbis-raw-coind2*:  
**assumes**  $\text{theta} \subseteq \text{Wretr } \text{theta}$  **and**  
 $\text{theta}^{-1} \subseteq \text{Wretr } (\text{theta}^{-1})$   
**shows**  $\text{theta} \subseteq \text{Wbis}$   
 $\langle \text{proof} \rangle$

**lemma** *WbisT-prefix*:  
 $\text{WbisT} \subseteq \text{WretrT } \text{WbisT}$   
 $\langle \text{proof} \rangle$



**lemma** *WbisT-sym: sym WbisT*

*<proof>*

**lemma** *WbisT-Sym:  $c \approx_{wT} d \implies d \approx_{wT} c$*

*<proof>*

**lemma** *WbisT-converse:*

*$((c,d) \in \text{theta}^{-1} \cup \text{WbisT}) = ((d,c) \in \text{theta} \cup \text{WbisT})$*

*<proof>*

**lemma**

*WbisT-matchC-MC:  $\bigwedge c d. c \approx_{wT} d \implies \text{matchC-MC } \text{WbisT } c d$*

**and**

*WbisT-matchT-MT:  $\bigwedge c d. c \approx_{wT} d \implies \text{matchT-MT } c d$*

*<proof>*

**lemmas** *WbisT-step = WbisT-matchC-MC WbisT-matchT-MT*

**lemma**

*WbisT-matchC-MC-rev:  $\bigwedge s t. s \approx_{wT} t \implies \text{matchC-MC } \text{WbisT } t s$*

**and**

*WbisT-matchT-MT-rev:  $\bigwedge s t. s \approx_{wT} t \implies \text{matchT-MT } t s$*

*<proof>*

**lemmas** *WbisT-step-rev = WbisT-matchC-MC-rev WbisT-matchT-MT-rev*

**lemma** *WbisT-coind:*

**assumes** *sym theta and  $\text{theta} \subseteq \text{WretrT } (\text{theta} \cup \text{WbisT})$*

**shows**  *$\text{theta} \subseteq \text{WbisT}$*

*<proof>*

**lemma** *WbisT-raw-coind:*

**assumes** *sym theta and  $\text{theta} \subseteq \text{WretrT } \text{theta}$*

**shows**  *$\text{theta} \subseteq \text{WbisT}$*

*<proof>*

**lemma** *WbisT-coind2:*

**assumes**  *$\text{theta} \subseteq \text{WretrT } (\text{theta} \cup \text{WbisT})$  and*

*$\text{theta}^{-1} \subseteq \text{WretrT } ((\text{theta}^{-1}) \cup \text{WbisT})$*

**shows**  *$\text{theta} \subseteq \text{WbisT}$*

*<proof>*

**lemma** *WbisT-raw-coind2:*

**assumes**  *$\text{theta} \subseteq \text{WretrT } \text{theta}$  and*

*$\text{theta}^{-1} \subseteq \text{WretrT } (\text{theta}^{-1})$*

**shows**  *$\text{theta} \subseteq \text{WbisT}$*

*<proof>*

**lemma** *WbisT-coinduct*[consumes 1, case-names sym cont term]:

**assumes**  $\varphi: \varphi\ c\ d$   
**assumes**  $S: \bigwedge c\ d. \varphi\ c\ d \implies \varphi\ d\ c$   
**assumes**  $C: \bigwedge c\ s\ d\ t\ c'\ s'.$   
 $\llbracket \varphi\ c\ d ; s \approx t ; (c, s) \rightarrow c\ (c', s') \rrbracket \implies \exists d'\ t'. (d, t) \rightarrow^* c\ (d', t') \wedge s' \approx t' \wedge$   
 $(\varphi\ c'\ d' \vee c' \approx_w T\ d')$   
**assumes**  $T: \bigwedge c\ s\ d\ t\ s'.$   
 $\llbracket \varphi\ c\ d ; s \approx t ; (c, s) \rightarrow t\ s' \rrbracket \implies \exists t'. (d, t) \rightarrow^* t' \wedge s' \approx t'$   
**shows**  $c \approx_w T\ d$   
 $\langle proof \rangle$

**lemma** *BisT-prefix*:

$BisT \subseteq RetrT\ BisT$

$\langle proof \rangle$

**lemma** *BisT-sym*:  $sym\ BisT$

$\langle proof \rangle$

**lemma** *BisT-Sym*:  $c \approx T\ d \implies d \approx T\ c$

$\langle proof \rangle$

**lemma** *BisT-converse*:

$((c, d) \in \widehat{theta^{-1}} \cup BisT) = ((d, c) \in theta \cup BisT)$

$\langle proof \rangle$

**lemma**

*BisT-matchC-TMC*:  $\bigwedge c\ d. c \approx T\ d \implies matchC-TMC\ BisT\ c\ d$

**and**

*BisT-matchT-TMT*:  $\bigwedge c\ d. c \approx T\ d \implies matchT-TMT\ c\ d$

$\langle proof \rangle$

**lemmas**  $BisT-step = BisT-matchC-TMC\ BisT-matchT-TMT$

**lemma**

*BisT-matchC-TMC-rev*:  $\bigwedge c\ d. c \approx T\ d \implies matchC-TMC\ BisT\ d\ c$

**and**

*BisT-matchT-TMT-rev*:  $\bigwedge c\ d. c \approx T\ d \implies matchT-TMT\ d\ c$

$\langle proof \rangle$

**lemmas**  $BisT-step-rev = BisT-matchC-TMC-rev\ BisT-matchT-TMT-rev$

**lemma** *BisT-coind*:

**assumes** *sym theta* **and**  $theta \subseteq RetrT\ (theta \cup BisT)$

**shows**  $theta \subseteq BisT$

$\langle proof \rangle$

**lemma** *BisT-raw-coind*:

**assumes** *sym theta* **and**  $theta \subseteq RetrT\ theta$

**shows**  $\theta \subseteq \text{BisT}$   
*<proof>*

**lemma** *BisT-coind2*:  
**assumes**  $\theta \subseteq \text{RetrT } (\theta \cup \text{BisT})$  **and**  
 $\theta^{-1} \subseteq \text{RetrT } ((\theta^{-1}) \cup \text{BisT})$   
**shows**  $\theta \subseteq \text{BisT}$   
*<proof>*

**lemma** *BisT-raw-coind2*:  
**assumes**  $\theta \subseteq \text{RetrT } \theta$  **and**  
 $\theta^{-1} \subseteq \text{RetrT } (\theta^{-1})$   
**shows**  $\theta \subseteq \text{BisT}$   
*<proof>*

Inclusions between bisimilarities:

**lemma** *match-imp[simp]*:  
 $\bigwedge \theta \ c1 \ c2. \text{matchC-C } \theta \ c1 \ c2 \implies \text{matchC-ZOC } \theta \ c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchC-ZOC } \theta \ c1 \ c2 \implies \text{matchC-ZO } \theta \ c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchC-ZOC } \theta \ c1 \ c2 \implies \text{matchC-MC } \theta \ c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchC-ZO } \theta \ c1 \ c2 \implies \text{matchC-M } \theta \ c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchC-MC } \theta \ c1 \ c2 \implies \text{matchC-M } \theta \ c1 \ c2$   
  
 $\bigwedge \ c1 \ c2. \text{matchT-T } c1 \ c2 \implies \text{matchT-ZO } c1 \ c2$   
 $\bigwedge \ c1 \ c2. \text{matchT-T } c1 \ c2 \implies \text{matchT-MT } c1 \ c2$   
 $\bigwedge \ c1 \ c2. \text{matchT-ZO } c1 \ c2 \implies \text{matchT-M } c1 \ c2$   
 $\bigwedge \ c1 \ c2. \text{matchT-MT } c1 \ c2 \implies \text{matchT-M } c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchC-MC } \theta \ c1 \ c2 \implies \text{matchC-TMC } \theta \ c1 \ c2$   
 $\bigwedge \theta \ c1 \ c2. \text{matchT-MT } c1 \ c2 \implies \text{matchT-TMT } c1 \ c2$   
*<proof>*

**lemma** *Retr-incl*:  
 $\bigwedge \theta. \text{Sretr } \theta \subseteq \text{ZOretrT } \theta$   
 $\bigwedge \theta. \text{ZOretrT } \theta \subseteq \text{ZOretr } \theta$   
 $\bigwedge \theta. \text{ZOretrT } \theta \subseteq \text{WretrT } \theta$   
 $\bigwedge \theta. \text{ZOretr } \theta \subseteq \text{Wretr } \theta$

$\bigwedge \text{theta}. \text{Wretr}T \text{ theta} \subseteq \text{Wretr} \text{ theta}$

$\bigwedge \text{theta}. \text{Wretr}T \text{ theta} \subseteq \text{Retr}T \text{ theta}$   
 $\langle \text{proof} \rangle$

**lemma** *bis-incl*:  
 $S\text{bis} \subseteq Z\text{Obis}T$

$Z\text{Obis}T \subseteq Z\text{Obis}$

$Z\text{Obis}T \subseteq W\text{bis}T$

$Z\text{Obis} \subseteq W\text{bis}$

$W\text{bis}T \subseteq W\text{bis}$

$W\text{bis}T \subseteq \text{Bis}T$   
 $\langle \text{proof} \rangle$

**lemma** *bis-imp[simp]*:  
 $\bigwedge c1 c2. c1 \approx_s c2 \implies c1 \approx_{01T} c2$

$\bigwedge c1 c2. c1 \approx_{01T} c2 \implies c1 \approx_{01} c2$

$\bigwedge c1 c2. c1 \approx_{01T} c2 \implies c1 \approx_{wT} c2$

$\bigwedge c1 c2. c1 \approx_{01} c2 \implies c1 \approx_w c2$

$\bigwedge c1 c2. c1 \approx_{wT} c2 \implies c1 \approx_w c2$

$\bigwedge c1 c2. c1 \approx_{wT} c2 \implies c1 \approx_T c2$   
 $\langle \text{proof} \rangle$

Self-isomorphism implies strong bisimilarity:

**lemma** *iso-Sbis[simp]*:  
**assumes** *iso* *c*  
**shows**  $c \approx_s c$   
 $\langle \text{proof} \rangle$

0-Self-isomorphism implies weak T 0-bisimilarity:

**lemma** *iso0-Sbis[simp]*:  
**assumes** *iso0* *c*  
**shows**  $c \approx_T c$   
 $\langle \text{proof} \rangle$

**end**

end

## 5 Compositionality of the during-execution security notions

**theory** *Compositionality* **imports** *During-Execution* **begin**

**context** *PL-Indis*  
**begin**

### 5.1 Discreetness versus language constructs:

**theorem** *discr-Atm[simp]*:  
 $\text{discr } (\text{Atm } atm) = \text{presAtm } atm$   
*<proof>*

**theorem** *discr-If[simp]*:  
**assumes** *discr c1 and discr c2*  
**shows** *discr (If tst c1 c2)*  
*<proof>*

**theorem** *discr-Seq[simp]*:  
**assumes** \*: *discr c1 and \*\*: discr c2*  
**shows** *discr (c1 ;; c2)*  
*<proof>*

**theorem** *discr-While[simp]*:  
**assumes** *discr c*  
**shows** *discr (While tst c)*  
*<proof>*

**theorem** *discr-Par[simp]*:  
**assumes** \*: *discr c1 and \*\*: discr c2*  
**shows** *discr (Par c1 c2)*  
*<proof>*

### 5.2 Discreetness versus language constructs:

**theorem** *discr0-Atm[simp]*:  
 $\text{discr0 } (\text{Atm } atm) = \text{presAtm } atm$   
*<proof>*

**theorem** *discr0-If[simp]*:  
**assumes** *discr0 c1 and discr0 c2*  
**shows** *discr0 (If tst c1 c2)*

*<proof>*

**theorem** *discr0-Seq[simp]*:  
**assumes** \*: *discr0 c1* **and** \*\*: *discr0 c2*  
**shows** *discr0 (c1 ;; c2)*  
*<proof>*

**theorem** *discr0-While[simp]*:  
**assumes** *discr0 c*  
**shows** *discr0 (While tst c)*  
*<proof>*

**theorem** *discr0-Par[simp]*:  
**assumes** \*: *discr0 c1* **and** \*\*: *discr0 c2*  
**shows** *discr0 (Par c1 c2)*  
*<proof>*

### 5.3 Self-Isomorphism versus language constructs:

**theorem** *siso-Atm[simp]*:  
*siso (Atm atm) = compatAtm atm*  
*<proof>*

**theorem** *siso-If[simp]*:  
**assumes** *compatTst tst* **and** *siso c1* **and** *siso c2*  
**shows** *siso (If tst c1 c2)*  
*<proof>*

**theorem** *siso-Seq[simp]*:  
**assumes** \*: *siso c1* **and** \*\*: *siso c2*  
**shows** *siso (c1 ;; c2)*  
*<proof>*

**theorem** *siso-While[simp]*:  
**assumes** *compatTst tst* **and** *siso c*  
**shows** *siso (While tst c)*  
*<proof>*

**theorem** *siso-Par[simp]*:  
**assumes** \*: *siso c1* **and** \*\*: *siso c2*  
**shows** *siso (Par c1 c2)*  
*<proof>*

### 5.4 Self-Isomorphism versus language constructs:

**theorem** *siso0-Atm[simp]*:  
*siso0 (Atm atm) = compatAtm atm*  
*<proof>*

**theorem** *siso0-If[simp]*:

**assumes** *compatTst tst* **and** *siso0 c1* **and** *siso0 c2*  
**shows** *siso0 (If tst c1 c2)*  
 $\langle$ *proof* $\rangle$

**theorem** *siso0-Seq[simp]*:  
**assumes** \*: *siso0 c1* **and** \*\*: *siso0 c2*  
**shows** *siso0 (c1 ;; c2)*  
 $\langle$ *proof* $\rangle$

**theorem** *siso0-While[simp]*:  
**assumes** *compatTst tst* **and** *siso0 c*  
**shows** *siso0 (While tst c)*  
 $\langle$ *proof* $\rangle$

**theorem** *siso0-Par[simp]*:  
**assumes** \*: *siso0 c1* **and** \*\*: *siso0 c2*  
**shows** *siso0 (Par c1 c2)*  
 $\langle$ *proof* $\rangle$

## 5.5 Strong bisimilarity versus language constructs

Atomic commands:

**definition** *thetaAtm* **where**  
*thetaAtm atm*  $\equiv \{(Atm\ atm, Atm\ atm)\}$

**lemma** *thetaAtm-sym*:  
*sym (thetaAtm atm)*  
 $\langle$ *proof* $\rangle$

**lemma** *thetaAtm-Sretr*:  
**assumes** *compatAtm atm*  
**shows** *thetaAtm atm*  $\subseteq$  *Sretr (thetaAtm atm)*  
 $\langle$ *proof* $\rangle$

**lemma** *thetaAtm-Sbis*:  
**assumes** *compatAtm atm*  
**shows** *thetaAtm atm*  $\subseteq$  *Sbis*  
 $\langle$ *proof* $\rangle$

**theorem** *Atm-Sbis[simp]*:  
**assumes** *compatAtm atm*  
**shows** *Atm atm*  $\approx_s$  *Atm atm*  
 $\langle$ *proof* $\rangle$

Sequential composition:

**definition** *thetaSeq* **where**  
*thetaSeq*  $\equiv$   
 $\{(c1\ ;;\ c2, d1\ ;;\ d2) \mid c1\ c2\ d1\ d2. c1\ \approx_s\ d1\ \wedge\ c2\ \approx_s\ d2\}$

**lemma** *thetaSeq-sym*:

*sym thetaSeq*

*<proof>*

**lemma** *thetaSeq-Sretr*:

*thetaSeq*  $\subseteq$  *Sretr* (*thetaSeq Un Sbis*)

*<proof>*

**lemma** *thetaSeq-Sbis*:

*thetaSeq*  $\subseteq$  *Sbis*

*<proof>*

**theorem** *Seq-Sbis[simp]*:

**assumes** *c1*  $\approx_s$  *d1* **and** *c2*  $\approx_s$  *d2*

**shows** *c1* ;; *c2*  $\approx_s$  *d1* ;; *d2*

*<proof>*

Conditional:

**definition** *thetaIf* **where**

*thetaIf*  $\equiv$

$\{(If\ tst\ c1\ c2,\ If\ tst\ d1\ d2) \mid tst\ c1\ c2\ d1\ d2.\ compatTst\ tst \wedge c1 \approx_s d1 \wedge c2 \approx_s d2\}$

**lemma** *thetaIf-sym*:

*sym thetaIf*

*<proof>*

**lemma** *thetaIf-Sretr*:

*thetaIf*  $\subseteq$  *Sretr* (*thetaIf Un Sbis*)

*<proof>*

**lemma** *thetaIf-Sbis*:

*thetaIf*  $\subseteq$  *Sbis*

*<proof>*

**theorem** *If-Sbis[simp]*:

**assumes** *compatTst tst* **and** *c1*  $\approx_s$  *d1* **and** *c2*  $\approx_s$  *d2*

**shows** *If tst c1 c2*  $\approx_s$  *If tst d1 d2*

*<proof>*

While loop:

**definition** *thetaWhile* **where**

*thetaWhile*  $\equiv$

$\{(While\ tst\ c,\ While\ tst\ d) \mid tst\ c\ d.\ compatTst\ tst \wedge c \approx_s d\} Un$   
 $\{(c1\ ;;\ (While\ tst\ c),\ d1\ ;;\ (While\ tst\ d)) \mid tst\ c1\ d1\ c\ d.\ compatTst\ tst \wedge c1 \approx_s d1 \wedge c \approx_s d\}$

**lemma** *thetaWhile-sym*:

*sym thetaWhile*



*<proof>*

**lemma** *thetaWhile-Sretr*:

*thetaWhile*  $\subseteq$  *Sretr* (*thetaWhile Un Sbis*)

*<proof>*

**lemma** *thetaWhile-Sbis*:

*thetaWhile*  $\subseteq$  *Sbis*

*<proof>*

**theorem** *While-Sbis[simp]*:

**assumes** *compatTst tst* **and**  $c \approx_s d$

**shows** *While tst c*  $\approx_s$  *While tst d*

*<proof>*

Parallel composition:

**definition** *thetaPar* **where**

*thetaPar*  $\equiv$

$\{(Par\ c1\ c2,\ Par\ d1\ d2) \mid c1\ c2\ d1\ d2.\ c1 \approx_s d1 \wedge c2 \approx_s d2\}$

**lemma** *thetaPar-sym*:

*sym thetaPar*

*<proof>*

**lemma** *thetaPar-Sretr*:

*thetaPar*  $\subseteq$  *Sretr* (*thetaPar Un Sbis*)

*<proof>*

**lemma** *thetaPar-Sbis*:

*thetaPar*  $\subseteq$  *Sbis*

*<proof>*

**theorem** *Par-Sbis[simp]*:

**assumes**  $c1 \approx_s d1$  **and**  $c2 \approx_s d2$

**shows** *Par c1 c2*  $\approx_s$  *Par d1 d2*

*<proof>*

### 5.5.1 01T-bisimilarity versus language constructs

Atomic commands:

**theorem** *Atm-ZObisT*:

**assumes** *compatAtm atm*

**shows** *Atm atm*  $\approx_{01T}$  *Atm atm*

*<proof>*

Sequential composition:

**definition** *thetaSeqZOT* **where**

*thetaSeqZOT*  $\equiv$

$\{(c1 ;; c2, d1 ;; d2) \mid c1 \ c2 \ d1 \ d2. \ c1 \approx_{01T} \ d1 \ \wedge \ c2 \approx_{01T} \ d2\}$

**lemma** *thetaSeqZOT-sym*:  
*sym thetaSeqZOT*  
(proof)

**lemma** *thetaSeqZOT-ZOretrT*:  
*thetaSeqZOT*  $\subseteq$  *ZOretrT* (*thetaSeqZOT Un ZObisT*)  
(proof)

**lemma** *thetaSeqZOT-ZObisT*:  
*thetaSeqZOT*  $\subseteq$  *ZObisT*  
(proof)

**theorem** *Seq-ZObisT[simp]*:  
**assumes** *c1*  $\approx_{01T}$  *d1* **and** *c2*  $\approx_{01T}$  *d2*  
**shows** *c1* ;; *c2*  $\approx_{01T}$  *d1* ;; *d2*  
(proof)

Conditional:

**definition** *thetaIfZOT* **where**  
*thetaIfZOT*  $\equiv$   
 $\{(If \ tst \ c1 \ c2, \ If \ tst \ d1 \ d2) \mid \ tst \ c1 \ c2 \ d1 \ d2. \ compatTst \ tst \ \wedge \ c1 \approx_{01T} \ d1 \ \wedge \ c2 \approx_{01T} \ d2\}$

**lemma** *thetaIfZOT-sym*:  
*sym thetaIfZOT*  
(proof)

**lemma** *thetaIfZOT-ZOretrT*:  
*thetaIfZOT*  $\subseteq$  *ZOretrT* (*thetaIfZOT Un ZObisT*)  
(proof)

**lemma** *thetaIfZOT-ZObisT*:  
*thetaIfZOT*  $\subseteq$  *ZObisT*  
(proof)

**theorem** *If-ZObisT[simp]*:  
**assumes** *compatTst* *tst* **and** *c1*  $\approx_{01T}$  *d1* **and** *c2*  $\approx_{01T}$  *d2*  
**shows** *If* *tst* *c1* *c2*  $\approx_{01T}$  *If* *tst* *d1* *d2*  
(proof)

While loop:

**definition** *thetaWhileZOT* **where**  
*thetaWhileZOT*  $\equiv$   
 $\{(While \ tst \ c, \ While \ tst \ d) \mid \ tst \ c \ d. \ compatTst \ tst \ \wedge \ c \approx_{01T} \ d\} \ Un$   
 $\{(c1 ;; (While \ tst \ c), \ d1 ;; (While \ tst \ d)) \mid \ tst \ c1 \ d1 \ c \ d. \ compatTst \ tst \ \wedge \ c1 \approx_{01T} \ d1 \ \wedge \ c \approx_{01T} \ d\}$

**lemma** *thetaWhileZOT-sym*:  
*sym thetaWhileZOT*  
 ⟨*proof*⟩

**lemma** *thetaWhileZOT-ZOretrT*:  
*thetaWhileZOT*  $\subseteq$  *ZOretrT* (*thetaWhileZOT Un ZObisT*)  
 ⟨*proof*⟩

**lemma** *thetaWhileZOT-ZObisT*:  
*thetaWhileZOT*  $\subseteq$  *ZObisT*  
 ⟨*proof*⟩

**theorem** *While-ZObisT[simp]*:  
**assumes** *compatTst tst* **and**  $c \approx_{01T} d$   
**shows** *While tst c*  $\approx_{01T}$  *While tst d*  
 ⟨*proof*⟩

Parallel composition:

**definition** *thetaParZOT* **where**  
*thetaParZOT*  $\equiv$   
 $\{(Par\ c1\ c2, Par\ d1\ d2) \mid c1\ c2\ d1\ d2. c1 \approx_{01T} d1 \wedge c2 \approx_{01T} d2\}$

**lemma** *thetaParZOT-sym*:  
*sym thetaParZOT*  
 ⟨*proof*⟩

**lemma** *thetaParZOT-ZOretrT*:  
*thetaParZOT*  $\subseteq$  *ZOretrT* (*thetaParZOT Un ZObisT*)  
 ⟨*proof*⟩

**lemma** *thetaParZOT-ZObisT*:  
*thetaParZOT*  $\subseteq$  *ZObisT*  
 ⟨*proof*⟩

**theorem** *Par-ZObisT[simp]*:  
**assumes**  $c1 \approx_{01T} d1$  **and**  $c2 \approx_{01T} d2$   
**shows** *Par c1 c2*  $\approx_{01T}$  *Par d1 d2*  
 ⟨*proof*⟩

## 5.5.2 01-bisimilarity versus language constructs

Discreetness:

**theorem** *discr-ZObis[simp]*:  
**assumes**  $*$ : *discr c* **and**  $**$ : *discr d*  
**shows**  $c \approx_{01} d$   
 ⟨*proof*⟩

Atomic commands:

**theorem** *Atm-ZObis[simp]*:

**assumes** *compatAtm atm*  
**shows**  $Atm\ atm \approx_{01}\ Atm\ atm$   
 $\langle proof \rangle$

Sequential composition:

**definition** *thetaSeqZO* where  
 $thetaSeqZO \equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1\ c2\ d1\ d2. c1 \approx_{01T}\ d1 \wedge c2 \approx_{01}\ d2\}$

**lemma** *thetaSeqZO-sym*:  
 $sym\ thetaSeqZO$   
 $\langle proof \rangle$

**lemma** *thetaSeqZO-ZOretr*:  
 $thetaSeqZO \subseteq ZOretr\ (thetaSeqZO\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaSeqZO-ZObis*:  
 $thetaSeqZO \subseteq ZObis$   
 $\langle proof \rangle$

**theorem** *Seq-ZObisT-ZObis[simp]*:  
**assumes**  $c1 \approx_{01T}\ d1$  **and**  $c2 \approx_{01}\ d2$   
**shows**  $c1 ;; c2 \approx_{01}\ d1 ;; d2$   
 $\langle proof \rangle$

**theorem** *Seq-iso-ZObis[simp]*:  
**assumes** *iso e* **and**  $c2 \approx_{01}\ d2$   
**shows**  $e ;; c2 \approx_{01}\ e ;; d2$   
 $\langle proof \rangle$

**definition** *thetaSeqZOD* where  
 $thetaSeqZOD \equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1\ c2\ d1\ d2. c1 \approx_{01}\ d1 \wedge discr\ c2 \wedge discr\ d2\}$

**lemma** *thetaSeqZOD-sym*:  
 $sym\ thetaSeqZOD$   
 $\langle proof \rangle$

**lemma** *thetaSeqZOD-ZOretr*:  
 $thetaSeqZOD \subseteq ZOretr\ (thetaSeqZOD\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaSeqZOD-ZObis*:  
 $thetaSeqZOD \subseteq ZObis$   
 $\langle proof \rangle$

**theorem** *Seq-ZObis-discr[simp]*:  
**assumes**  $c1 \approx_{01} d1$  **and**  $discr\ c2$  **and**  $discr\ d2$   
**shows**  $c1 ;; c2 \approx_{01} d1 ;; d2$   
 $\langle proof \rangle$

Conditional:

**definition** *thetaIfZO* **where**  
 $thetaIfZO \equiv$   
 $\{(If\ tst\ c1\ c2,\ If\ tst\ d1\ d2) \mid tst\ c1\ c2\ d1\ d2.\ compatTst\ tst \wedge c1 \approx_{01} d1 \wedge c2 \approx_{01} d2\}$

**lemma** *thetaIfZO-sym*:  
 $sym\ thetaIfZO$   
 $\langle proof \rangle$

**lemma** *thetaIfZO-ZOretr*:  
 $thetaIfZO \subseteq ZOretr\ (thetaIfZO\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaIfZO-ZObis*:  
 $thetaIfZO \subseteq ZObis$   
 $\langle proof \rangle$

**theorem** *If-ZObis[simp]*:  
**assumes**  $compatTst\ tst$  **and**  $c1 \approx_{01} d1$  **and**  $c2 \approx_{01} d2$   
**shows**  $If\ tst\ c1\ c2 \approx_{01} If\ tst\ d1\ d2$   
 $\langle proof \rangle$

While loop:

01-bisimilarity does not interact with / preserve the While construct in any interesting way.

Parallel composition:

**definition** *thetaParZOL1* **where**  
 $thetaParZOL1 \equiv$   
 $\{(Par\ c1\ c2,\ d) \mid c1\ c2\ d.\ c1 \approx_{01} d \wedge discr\ c2\}$

**lemma** *thetaParZOL1-ZOretr*:  
 $thetaParZOL1 \subseteq ZOretr\ (thetaParZOL1\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaParZOL1-converse-ZOretr*:  
 $thetaParZOL1 \hat{\sim}^{-1} \subseteq ZOretr\ (thetaParZOL1 \hat{\sim}^{-1}\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaParZOL1-ZObis*:  
 $thetaParZOL1 \subseteq ZObis$   
 $\langle proof \rangle$

**theorem** *Par-ZObis-discrL1*[simp]:  
**assumes**  $c1 \approx 01 d$  **and**  $discr\ c2$   
**shows**  $Par\ c1\ c2 \approx 01 d$   
 $\langle proof \rangle$

**theorem** *Par-ZObis-discrR1*[simp]:  
**assumes**  $c \approx 01 d1$  **and**  $discr\ d2$   
**shows**  $c \approx 01 Par\ d1\ d2$   
 $\langle proof \rangle$

**definition** *thetaParZOL2* **where**  
 $thetaParZOL2 \equiv$   
 $\{(Par\ c1\ c2, d) \mid c1\ c2\ d.\ discr\ c1 \wedge c2 \approx 01 d\}$

**lemma** *thetaParZOL2-ZOretr*:  
 $thetaParZOL2 \subseteq ZOretr\ (thetaParZOL2\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaParZOL2-converse-ZOretr*:  
 $thetaParZOL2\ ^{-1} \subseteq ZOretr\ (thetaParZOL2\ ^{-1}\ Un\ ZObis)$   
 $\langle proof \rangle$

**lemma** *thetaParZOL2-ZObis*:  
 $thetaParZOL2 \subseteq ZObis$   
 $\langle proof \rangle$

**theorem** *Par-ZObis-discrL2*[simp]:  
**assumes**  $c2 \approx 01 d$  **and**  $discr\ c1$   
**shows**  $Par\ c1\ c2 \approx 01 d$   
 $\langle proof \rangle$

**theorem** *Par-ZObis-discrR2*[simp]:  
**assumes**  $c \approx 01 d2$  **and**  $discr\ d1$   
**shows**  $c \approx 01 Par\ d1\ d2$   
 $\langle proof \rangle$

**definition** *thetaParZO* **where**  
 $thetaParZO \equiv$   
 $\{(Par\ c1\ c2, Par\ d1\ d2) \mid c1\ c2\ d1\ d2.\ c1 \approx 01 d1 \wedge c2 \approx 01 d2\}$

**lemma** *thetaParZO-sym*:  
 $sym\ thetaParZO$   
 $\langle proof \rangle$

**lemma** *thetaParZO-ZOretr*:  
*thetaParZO*  $\subseteq$  *ZOretr* (*thetaParZO* *Un* *ZObis*)  
 ⟨*proof*⟩

**lemma** *thetaParZO-ZObis*:  
*thetaParZO*  $\subseteq$  *ZObis*  
 ⟨*proof*⟩

**theorem** *Par-ZObis[simp]*:  
**assumes** *c1*  $\approx_{01}$  *d1* **and** *c2*  $\approx_{01}$  *d2*  
**shows** *Par c1 c2*  $\approx_{01}$  *Par d1 d2*  
 ⟨*proof*⟩

### 5.5.3 WT-bisimilarity versus language constructs

Discreetness:

**theorem** *noWhile-discr-WbisT[simp]*:  
**assumes** *noWhile c1* **and** *noWhile c2*  
**and** *discr c1* **and** *discr c2*  
**shows** *c1*  $\approx_{wT}$  *c2*  
 ⟨*proof*⟩

Atomic commands:

**theorem** *Atm-WbisT*:  
**assumes** *compatAtm atm*  
**shows** *Atm atm*  $\approx_{wT}$  *Atm atm*  
 ⟨*proof*⟩

Sequential composition:

**definition** *thetaSeqWT* **where**  
*thetaSeqWT*  $\equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1 c2 d1 d2. c1 \approx_{wT} d1 \wedge c2 \approx_{wT} d2\}$

**lemma** *thetaSeqWT-sym*:  
*sym thetaSeqWT*  
 ⟨*proof*⟩

**lemma** *thetaSeqWT-WretrT*:  
*thetaSeqWT*  $\subseteq$  *WretrT* (*thetaSeqWT* *Un* *WbisT*)  
 ⟨*proof*⟩

**lemma** *thetaSeqWT-WbisT*:  
*thetaSeqWT*  $\subseteq$  *WbisT*  
 ⟨*proof*⟩

**theorem** *Seq-WbisT[simp]*:  
**assumes** *c1*  $\approx_{wT}$  *d1* **and** *c2*  $\approx_{wT}$  *d2*  
**shows** *c1 ;; c2*  $\approx_{wT}$  *d1 ;; d2*

*<proof>*

Conditional:

**definition** *thetaIfWT* **where**

$thetaIfWT \equiv \{(If\ tst\ c1\ c2, If\ tst\ d1\ d2) \mid tst\ c1\ c2\ d1\ d2.\ compatTst\ tst \wedge c1 \approx_{wT} d1 \wedge c2 \approx_{wT} d2\}$

**lemma** *thetaIfWT-sym*:

*sym thetaIfWT*

*<proof>*

**lemma** *thetaIfWT-WretrT*:

$thetaIfWT \subseteq WretrT\ (thetaIfWT\ Un\ WbisT)$

*<proof>*

**lemma** *thetaIfWT-WbisT*:

$thetaIfWT \subseteq WbisT$

*<proof>*

**theorem** *If-WbisT[simp]*:

**assumes** *compatTst tst* **and**  $c1 \approx_{wT} d1$  **and**  $c2 \approx_{wT} d2$

**shows**  $If\ tst\ c1\ c2 \approx_{wT} If\ tst\ d1\ d2$

*<proof>*

While loop:

**definition** *thetaWhileW* **where**

$thetaWhileW \equiv \{(While\ tst\ c, While\ tst\ d) \mid tst\ c\ d.\ compatTst\ tst \wedge c \approx_{wT} d\}\ Un\ \{(c1\ ;;\ (While\ tst\ c),\ d1\ ;;\ (While\ tst\ d)) \mid tst\ c1\ d1\ c\ d.\ compatTst\ tst \wedge c1 \approx_{wT} d1 \wedge c \approx_{wT} d\}$

**lemma** *thetaWhileW-sym*:

*sym thetaWhileW*

*<proof>*

**lemma** *thetaWhileW-WretrT*:

$thetaWhileW \subseteq WretrT\ (thetaWhileW\ Un\ WbisT)$

*<proof>*

**lemma** *thetaWhileW-WbisT*:

$thetaWhileW \subseteq WbisT$

*<proof>*

**theorem** *While-WbisT[simp]*:

**assumes** *compatTst tst* **and**  $c \approx_{wT} d$

**shows**  $While\ tst\ c \approx_{wT} While\ tst\ d$

*<proof>*

Parallel composition:



**definition** *thetaParWT* where

$$\begin{aligned} & \textit{thetaParWT} \equiv \\ & \{(Par\ c1\ c2, Par\ d1\ d2) \mid c1\ c2\ d1\ d2. c1 \approx_{wT} d1 \wedge c2 \approx_{wT} d2\} \end{aligned}$$

**lemma** *thetaParWT-sym*:

$$\begin{aligned} & \textit{sym\ thetaParWT} \\ & \langle \textit{proof} \rangle \end{aligned}$$

**lemma** *thetaParWT-WretrT*:

$$\begin{aligned} & \textit{thetaParWT} \subseteq \textit{WretrT}\ (\textit{thetaParWT}\ \textit{Un}\ \textit{WbisT}) \\ & \langle \textit{proof} \rangle \end{aligned}$$

**lemma** *thetaParWT-WbisT*:

$$\begin{aligned} & \textit{thetaParWT} \subseteq \textit{WbisT} \\ & \langle \textit{proof} \rangle \end{aligned}$$

**theorem** *Par-WbisT[simp]*:

$$\begin{aligned} & \textit{assumes}\ c1 \approx_{wT} d1\ \textit{and}\ c2 \approx_{wT} d2 \\ & \textit{shows}\ Par\ c1\ c2 \approx_{wT} Par\ d1\ d2 \\ & \langle \textit{proof} \rangle \end{aligned}$$

#### 5.5.4 T-bisimilarity versus language constructs

T-Discreetness:

**definition** *thetaFDW0* where

$$\begin{aligned} & \textit{thetaFDW0} \equiv \\ & \{(c1, c2). \textit{discr0}\ c1 \wedge \textit{discr0}\ c2\} \end{aligned}$$

**lemma** *thetaFDW0-sym*:

$$\begin{aligned} & \textit{sym\ thetaFDW0} \\ & \langle \textit{proof} \rangle \end{aligned}$$

**lemma** *thetaFDW0-RetrT*:

$$\begin{aligned} & \textit{thetaFDW0} \subseteq \textit{RetrT}\ \textit{thetaFDW0} \\ & \langle \textit{proof} \rangle \end{aligned}$$

**lemma** *thetaFDW0-BisT*:

$$\begin{aligned} & \textit{thetaFDW0} \subseteq \textit{BisT} \\ & \langle \textit{proof} \rangle \end{aligned}$$

**theorem** *discr0-BisT[simp]*:

$$\begin{aligned} & \textit{assumes}\ \textit{discr0}\ c1\ \textit{and}\ \textit{discr0}\ c2 \\ & \textit{shows}\ c1 \approx_T c2 \\ & \langle \textit{proof} \rangle \end{aligned}$$

Atomic commands:

**theorem** *Atm-BisT*:

$$\begin{aligned} & \textit{assumes}\ \textit{compatAtm}\ atm \\ & \textit{shows}\ \textit{Atm}\ atm \approx_T \textit{Atm}\ atm \end{aligned}$$

*<proof>*

Sequential composition:

**definition** *thetaSeqTT* **where**

$thetaSeqTT \equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1 \ c2 \ d1 \ d2. \ c1 \approx_T \ d1 \ \wedge \ c2 \approx_T \ d2\}$

**lemma** *thetaSeqTT-sym*:

*sym thetaSeqTT*

*<proof>*

**lemma** *thetaSeqTT-RetrT*:

$thetaSeqTT \subseteq RetrT \ (thetaSeqTT \cup BisT)$

*<proof>*

**lemma** *thetaSeqTT-BisT*:

$thetaSeqTT \subseteq BisT$

*<proof>*

**theorem** *Seq-BisT[simp]*:

**assumes**  $c1 \approx_T \ d1$  **and**  $c2 \approx_T \ d2$

**shows**  $c1 ;; c2 \approx_T \ d1 ;; d2$

*<proof>*

Conditional:

**definition** *thetaIfTT* **where**

$thetaIfTT \equiv$   
 $\{(If \ tst \ c1 \ c2, \ If \ tst \ d1 \ d2) \mid \ tst \ c1 \ c2 \ d1 \ d2. \ compatTst \ tst \ \wedge \ c1 \approx_T \ d1 \ \wedge \ c2 \approx_T \ d2\}$

**lemma** *thetaIfTT-sym*:

*sym thetaIfTT*

*<proof>*

**lemma** *thetaIfTT-RetrT*:

$thetaIfTT \subseteq RetrT \ (thetaIfTT \cup BisT)$

*<proof>*

**lemma** *thetaIfTT-BisT*:

$thetaIfTT \subseteq BisT$

*<proof>*

**theorem** *If-BisT[simp]*:

**assumes**  $compatTst \ tst$  **and**  $c1 \approx_T \ d1$  **and**  $c2 \approx_T \ d2$

**shows**  $If \ tst \ c1 \ c2 \approx_T \ If \ tst \ d1 \ d2$

*<proof>*

While loop:

**definition** *thetaWhileW0* **where**

$\text{thetaWhile}W0 \equiv$   
 $\{(While\ tst\ c,\ While\ tst\ d) \mid \text{tst}\ c\ d.\ \text{compatTst}\ \text{tst} \wedge c \approx^T d\} \cup$   
 $\{(c1\ ;;\ (While\ tst\ c),\ d1\ ;;\ (While\ tst\ d)) \mid \text{tst}\ c1\ d1\ c\ d.\$   
 $\quad \text{compatTst}\ \text{tst} \wedge c1 \approx^T d1 \wedge c \approx^T d\}$

**lemma** *thetaWhileW0-sym*:

*sym thetaWhileW0*

*<proof>*

**lemma** *thetaWhileW0-RetrT*:

$\text{thetaWhile}W0 \subseteq \text{Retr}T\ (\text{thetaWhile}W0 \cup \text{Bis}T)$

*<proof>*

**lemma** *thetaWhileW0-BisT*:

$\text{thetaWhile}W0 \subseteq \text{Bis}T$

*<proof>*

**theorem** *While-BisT[simp]*:

**assumes**  $\text{compatTst}\ \text{tst}$  **and**  $c \approx^T d$

**shows**  $While\ \text{tst}\ c \approx^T While\ \text{tst}\ d$

*<proof>*

Parallel composition:

**definition** *thetaParTT* **where**

$\text{thetaPar}TT \equiv$

$\{(Par\ c1\ c2,\ Par\ d1\ d2) \mid c1\ c2\ d1\ d2.\ c1 \approx^T d1 \wedge c2 \approx^T d2\}$

**lemma** *thetaParTT-sym*:

*sym thetaParTT*

*<proof>*

**lemma** *thetaParTT-RetrT*:

$\text{thetaPar}TT \subseteq \text{Retr}T\ (\text{thetaPar}TT \cup \text{Bis}T)$

*<proof>*

**lemma** *thetaParTT-BisT*:

$\text{thetaPar}TT \subseteq \text{Bis}T$

*<proof>*

**theorem** *Par-BisT[simp]*:

**assumes**  $c1 \approx^T d1$  **and**  $c2 \approx^T d2$

**shows**  $Par\ c1\ c2 \approx^T Par\ d1\ d2$

*<proof>*

### 5.5.5 W-bisimilarity versus language constructs

Atomic commands:

**theorem** *Atm-Wbis[simp]*:

**assumes**  $\text{compatAtm}\ \text{atm}$

**shows**  $Atm\ atm \approx_w Atm\ atm$   
*<proof>*

Discreetness:

**theorem** *discr-Wbis[simp]*:  
**assumes** \*: *discr c* **and** \*\*: *discr d*  
**shows**  $c \approx_w d$   
*<proof>*

Sequential composition:

**definition** *thetaSeqW* **where**  
 $thetaSeqW \equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1\ c2\ d1\ d2. c1 \approx_{wT} d1 \wedge c2 \approx_w d2\}$

**lemma** *thetaSeqW-sym*:  
*sym thetaSeqW*  
*<proof>*

**lemma** *thetaSeqW-Wretr*:  
 $thetaSeqW \subseteq Wretr\ (thetaSeqW \cup Wbis)$   
*<proof>*

**lemma** *thetaSeqW-Wbis*:  
 $thetaSeqW \subseteq Wbis$   
*<proof>*

**theorem** *Seq-WbisT-Wbis[simp]*:  
**assumes**  $c1 \approx_{wT} d1$  **and**  $c2 \approx_w d2$   
**shows**  $c1 ;; c2 \approx_w d1 ;; d2$   
*<proof>*

**theorem** *Seq-iso-Wbis[simp]*:  
**assumes** *iso e* **and**  $c2 \approx_w d2$   
**shows**  $e ;; c2 \approx_w e ;; d2$   
*<proof>*

**definition** *thetaSeqWD* **where**  
 $thetaSeqWD \equiv$   
 $\{(c1 ;; c2, d1 ;; d2) \mid c1\ c2\ d1\ d2. c1 \approx_w d1 \wedge discr\ c2 \wedge discr\ d2\}$

**lemma** *thetaSeqWD-sym*:  
*sym thetaSeqWD*  
*<proof>*

**lemma** *thetaSeqWD-Wretr*:  
 $thetaSeqWD \subseteq Wretr\ (thetaSeqWD \cup Wbis)$   
*<proof>*

**lemma** *thetaSeqWD-Wbis*:

$thetaSeqWD \subseteq Wbis$

*<proof>*

**theorem** *Seq-Wbis-discr[simp]*:

**assumes**  $c1 \approx_w d1$  **and**  $discr\ c2$  **and**  $discr\ d2$

**shows**  $c1 ;; c2 \approx_w d1 ;; d2$

*<proof>*

Conditional:

**definition** *thetaIfW* **where**

$thetaIfW \equiv$

$\{(If\ tst\ c1\ c2,\ If\ tst\ d1\ d2) \mid tst\ c1\ c2\ d1\ d2.\ compatTst\ tst \wedge c1 \approx_w d1 \wedge c2 \approx_w d2\}$

**lemma** *thetaIfW-sym*:

$sym\ thetaIfW$

*<proof>*

**lemma** *thetaIfW-Wretr*:

$thetaIfW \subseteq Wretr\ (thetaIfW \cup Wbis)$

*<proof>*

**lemma** *thetaIfW-Wbis*:

$thetaIfW \subseteq Wbis$

*<proof>*

**theorem** *If-Wbis[simp]*:

**assumes**  $compatTst\ tst$  **and**  $c1 \approx_w d1$  **and**  $c2 \approx_w d2$

**shows**  $If\ tst\ c1\ c2 \approx_w If\ tst\ d1\ d2$

*<proof>*

While loop:

Again, w-bisimilarity does not interact with / preserve the While construct in any interesting way.

Parallel composition:

**definition** *thetaParWL1* **where**

$thetaParWL1 \equiv$

$\{(Par\ c1\ c2,\ d) \mid c1\ c2\ d.\ c1 \approx_w d \wedge discr\ c2\}$

**lemma** *thetaParWL1-Wretr*:

$thetaParWL1 \subseteq Wretr\ (thetaParWL1 \cup Wbis)$

*<proof>*

**lemma** *thetaParWL1-converse-Wretr*:

$thetaParWL1 \hat{\sim}^{-1} \subseteq Wretr\ (thetaParWL1 \hat{\sim}^{-1} \cup Wbis)$

*<proof>*

**lemma** *thetaParWL1-Wbis*:

*thetaParWL1*  $\subseteq$  *Wbis*

*<proof>*

**theorem** *Par-Wbis-discrL1[simp]*:

**assumes** *c1*  $\approx_w$  *d* **and** *discr c2*

**shows** *Par c1 c2*  $\approx_w$  *d*

*<proof>*

**theorem** *Par-Wbis-discrR1[simp]*:

**assumes** *c*  $\approx_w$  *d1* **and** *discr d2*

**shows** *c*  $\approx_w$  *Par d1 d2*

*<proof>*

**definition** *thetaParWL2* **where**

*thetaParWL2*  $\equiv$

$\{(Par\ c1\ c2,\ d) \mid c1\ c2\ d.\ discr\ c1 \wedge c2 \approx_w d\}$

**lemma** *thetaParWL2-Wretr*:

*thetaParWL2*  $\subseteq$  *Wretr* (*thetaParWL2*  $\cup$  *Wbis*)

*<proof>*

**lemma** *thetaParWL2-converse-Wretr*:

*thetaParWL2*  $\hat{-}1 \subseteq$  *Wretr* (*thetaParWL2*  $\hat{-}1 \cup$  *Wbis*)

*<proof>*

**lemma** *thetaParWL2-Wbis*:

*thetaParWL2*  $\subseteq$  *Wbis*

*<proof>*

**theorem** *Par-Wbis-discrL2[simp]*:

**assumes** *c2*  $\approx_w$  *d* **and** *discr c1*

**shows** *Par c1 c2*  $\approx_w$  *d*

*<proof>*

**theorem** *Par-Wbis-discrR2[simp]*:

**assumes** *c*  $\approx_w$  *d2* **and** *discr d1*

**shows** *c*  $\approx_w$  *Par d1 d2*

*<proof>*

**definition** *thetaParW* **where**

*thetaParW*  $\equiv$

$\{(Par\ c1\ c2,\ Par\ d1\ d2) \mid c1\ c2\ d1\ d2.\ c1 \approx_w d1 \wedge c2 \approx_w d2\}$

**lemma** *thetaParW-sym*:

*sym thetaParW*

*<proof>*

**lemma** *thetaParW-Wretr*:

*thetaParW*  $\subseteq$  *Wretr* (*thetaParW*  $\cup$  *Wbis*)

*<proof>*

**lemma** *thetaParW-Wbis*:

*thetaParW*  $\subseteq$  *Wbis*

*<proof>*

**theorem** *Par-Wbis[simp]*:

**assumes** *c1*  $\approx_w$  *d1* **and** *c2*  $\approx_w$  *d2*

**shows** *Par c1 c2*  $\approx_w$  *Par d1 d2*

*<proof>*

**end**

**end**

**theory** *Syntactic-Criteria*

**imports** *Compositionality*

**begin**

**context** *PL-Indis*

**begin**

**lemma** *noWhile[intro]*:

*noWhile* (*Atm atm*)

*noWhile c1*  $\implies$  *noWhile c2*  $\implies$  *noWhile* (*Seq c1 c2*)

*noWhile c1*  $\implies$  *noWhile c2*  $\implies$  *noWhile* (*If tst c1 c2*)

*noWhile c1*  $\implies$  *noWhile c2*  $\implies$  *noWhile* (*Par c1 c2*)

*<proof>*

**lemma** *discr[intro]*:

*presAtm atm*  $\implies$  *discr* (*Atm atm*)

*discr c1*  $\implies$  *discr c2*  $\implies$  *discr* (*Seq c1 c2*)

*discr c1*  $\implies$  *discr c2*  $\implies$  *discr* (*If tst c1 c2*)

*discr c*  $\implies$  *discr* (*While tst c*)

*discr c1*  $\implies$  *discr c2*  $\implies$  *discr* (*Par c1 c2*)

*<proof>*

**lemma** *siso[intro]*:

*compatAtm atm*  $\implies$  *siso* (*Atm atm*)

$siso\ c1 \implies siso\ c2 \implies siso\ (Seq\ c1\ c2)$   
 $compatTst\ tst \implies siso\ c1 \implies siso\ c2 \implies siso\ (If\ tst\ c1\ c2)$   
 $compatTst\ tst \implies siso\ c \implies siso\ (While\ tst\ c)$   
 $siso\ c1 \implies siso\ c2 \implies siso\ (Par\ c1\ c2)$   
 <proof>

**lemma** *Sbis*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_s\ Atm\ atm$   
 $c1 \approx_s\ c1 \implies c2 \approx_s\ c2 \implies Seq\ c1\ c2 \approx_s\ Seq\ c1\ c2$   
 $compatTst\ tst \implies c1 \approx_s\ c1 \implies c2 \approx_s\ c2 \implies If\ tst\ c1\ c2 \approx_s\ If\ tst\ c1\ c2$   
 $compatTst\ tst \implies c \approx_s\ c \implies While\ tst\ c \approx_s\ While\ tst\ c$   
 $c1 \approx_s\ c1 \implies c2 \approx_s\ c2 \implies Par\ c1\ c2 \approx_s\ Par\ c1\ c2$   
 <proof>

**lemma** *ZObisT*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_{01T}\ Atm\ atm$   
 $c1 \approx_{01T}\ c1 \implies c2 \approx_{01T}\ c2 \implies Seq\ c1\ c2 \approx_{01T}\ Seq\ c1\ c2$   
 $compatTst\ tst \implies c1 \approx_{01T}\ c1 \implies c2 \approx_{01T}\ c2 \implies If\ tst\ c1\ c2 \approx_{01T}\ If\ tst\ c1\ c2$   
 $compatTst\ tst \implies c \approx_{01T}\ c \implies While\ tst\ c \approx_{01T}\ While\ tst\ c$   
 $c1 \approx_{01T}\ c1 \implies c2 \approx_{01T}\ c2 \implies Par\ c1\ c2 \approx_{01T}\ Par\ c1\ c2$   
 <proof>

**lemma** *BisT*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_T\ Atm\ atm$   
 $c1 \approx_T\ c1 \implies c2 \approx_T\ c2 \implies Seq\ c1\ c2 \approx_T\ Seq\ c1\ c2$   
 $compatTst\ tst \implies c1 \approx_T\ c1 \implies c2 \approx_T\ c2 \implies If\ tst\ c1\ c2 \approx_T\ If\ tst\ c1\ c2$   
 $compatTst\ tst \implies c \approx_T\ c \implies While\ tst\ c \approx_T\ While\ tst\ c$   
 $c1 \approx_T\ c1 \implies c2 \approx_T\ c2 \implies Par\ c1\ c2 \approx_T\ Par\ c1\ c2$   
 <proof>

**lemma** *WbisT*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_{wT}\ Atm\ atm$   
 $c1 \approx_{wT}\ c1 \implies c2 \approx_{wT}\ c2 \implies Seq\ c1\ c2 \approx_{wT}\ Seq\ c1\ c2$   
 $compatTst\ tst \implies c1 \approx_{wT}\ c1 \implies c2 \approx_{wT}\ c2 \implies If\ tst\ c1\ c2 \approx_{wT}\ If\ tst\ c1\ c2$   
 $compatTst\ tst \implies c \approx_{wT}\ c \implies While\ tst\ c \approx_{wT}\ While\ tst\ c$   
 $c1 \approx_{wT}\ c1 \implies c2 \approx_{wT}\ c2 \implies Par\ c1\ c2 \approx_{wT}\ Par\ c1\ c2$   
 <proof>

**lemma** *ZObis*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_{01}\ Atm\ atm$   
 $c1 \approx_{01T}\ c1 \implies c2 \approx_{01}\ c2 \implies Seq\ c1\ c2 \approx_{01}\ Seq\ c1\ c2$   
 $c1 \approx_{01}\ c1 \implies discr\ c2 \implies Seq\ c1\ c2 \approx_{01}\ Seq\ c1\ c2$   
 $compatTst\ tst \implies c1 \approx_{01}\ c1 \implies c2 \approx_{01}\ c2 \implies If\ tst\ c1\ c2 \approx_{01}\ If\ tst\ c1\ c2$   
 $c1 \approx_{01}\ c1 \implies c2 \approx_{01}\ c2 \implies Par\ c1\ c2 \approx_{01}\ Par\ c1\ c2$   
 <proof>

**lemma** *Wbis*[intro]:

$compatAtm\ atm \implies Atm\ atm \approx_w\ Atm\ atm$



$c1 \approx_w T c1 \implies c2 \approx_w c2 \implies \text{Seq } c1 \ c2 \approx_w \text{Seq } c1 \ c2$   
 $c1 \approx_w c1 \implies \text{discr } c2 \implies \text{Seq } c1 \ c2 \approx_w \text{Seq } c1 \ c2$   
 $\text{compatTst } \text{tst} \implies c1 \approx_w c1 \implies c2 \approx_w c2 \implies \text{If } \text{tst } c1 \ c2 \approx_w \text{If } \text{tst } c1 \ c2$   
 $c1 \approx_w c1 \implies c2 \approx_w c2 \implies \text{Par } c1 \ c2 \approx_w \text{Par } c1 \ c2$   
 <proof>

**lemma** *discr-noWhile-WbisT*[intro]:  $\text{discr } c \implies \text{noWhile } c \implies c \approx_w T c$   
 <proof>

**lemma** *siso-ZObis*[intro]:  $\text{siso } c \implies c \approx_{01} c$   
 <proof>

**lemma** *WbisT-Wbis*[intro]:  $c \approx_w T c \implies c \approx_w c$   
 <proof>

**lemma** *ZObis-Wbis*[intro]:  $c \approx_{01} c \implies c \approx_w c$   
 <proof>

**lemma** *discr-BisT*[intro]:  $\text{discr } c \implies c \approx_T c$   
 <proof>

**lemma** *WbisT-BisT*[intro]:  $c \approx_w T c \implies c \approx_T c$   
 <proof>

**lemma** *ZObisT-ZObis*[intro]:  $c \approx_{01T} c \implies c \approx_{01} c$   
 <proof>

**lemma** *siso-ZObisT*[intro]:  $\text{siso } c \implies c \approx_{01T} c$   
 <proof>

**primrec** *SC-discr* **where**

$\text{SC-discr } (\text{Atm } \text{atm}) \quad \longleftrightarrow \text{presAtm } \text{atm}$   
 $| \text{SC-discr } (\text{Seq } c1 \ c2) \quad \longleftrightarrow \text{SC-discr } c1 \ \wedge \ \text{SC-discr } c2$   
 $| \text{SC-discr } (\text{If } \text{tst } c1 \ c2) \quad \longleftrightarrow \text{SC-discr } c1 \ \wedge \ \text{SC-discr } c2$   
 $| \text{SC-discr } (\text{While } \text{tst } c) \quad \longleftrightarrow \text{SC-discr } c$   
 $| \text{SC-discr } (\text{Par } c1 \ c2) \quad \longleftrightarrow \text{SC-discr } c1 \ \wedge \ \text{SC-discr } c2$

**primrec** *SC-siso* **where**

$\text{SC-siso } (\text{Atm } \text{atm}) \quad \longleftrightarrow \text{compatAtm } \text{atm}$   
 $| \text{SC-siso } (\text{Seq } c1 \ c2) \quad \longleftrightarrow \text{SC-siso } c1 \ \wedge \ \text{SC-siso } c2$   
 $| \text{SC-siso } (\text{If } \text{tst } c1 \ c2) \quad \longleftrightarrow \text{compatTst } \text{tst} \ \wedge \ \text{SC-siso } c1 \ \wedge \ \text{SC-siso } c2$   
 $| \text{SC-siso } (\text{While } \text{tst } c) \quad \longleftrightarrow \text{compatTst } \text{tst} \ \wedge \ \text{SC-siso } c$   
 $| \text{SC-siso } (\text{Par } c1 \ c2) \quad \longleftrightarrow \text{SC-siso } c1 \ \wedge \ \text{SC-siso } c2$

**primrec** *SC-WbisT* **where**

$\text{SC-WbisT } (\text{Atm } \text{atm}) \quad \longleftrightarrow \text{compatAtm } \text{atm}$

$$\begin{aligned}
| \text{SC-WbisT } (\text{Seq } c1 \ c2) & \longleftrightarrow (\text{SC-WbisT } c1 \wedge \text{SC-WbisT } c2) \vee \\
& (\text{noWhile } (\text{Seq } c1 \ c2) \wedge \text{SC-discr } (\text{Seq } c1 \ c2)) \vee \\
& \text{SC-siso } (\text{Seq } c1 \ c2) \\
| \text{SC-WbisT } (\text{If } \text{tst } c1 \ c2) & \longleftrightarrow (\text{if compatTst } \text{tst} \\
& \text{then } (\text{SC-WbisT } c1 \wedge \text{SC-WbisT } c2) \\
& \text{else } ((\text{noWhile } (\text{If } \text{tst } c1 \ c2) \wedge \text{SC-discr } (\text{If } \text{tst } c1 \ c2)) \vee \\
& \text{SC-siso } (\text{If } \text{tst } c1 \ c2))) \\
| \text{SC-WbisT } (\text{While } \text{tst } c) & \longleftrightarrow (\text{if compatTst } \text{tst} \\
& \text{then } \text{SC-WbisT } c \\
& \text{else } ((\text{noWhile } (\text{While } \text{tst } c) \wedge \text{SC-discr } (\text{While } \text{tst } c)) \vee \\
& \text{SC-siso } (\text{While } \text{tst } c))) \\
| \text{SC-WbisT } (\text{Par } c1 \ c2) & \longleftrightarrow (\text{SC-WbisT } c1 \wedge \text{SC-WbisT } c2) \vee \\
& (\text{noWhile } (\text{Par } c1 \ c2) \wedge \text{SC-discr } (\text{Par } c1 \ c2)) \vee \\
& \text{SC-siso } (\text{Par } c1 \ c2)
\end{aligned}$$

**primrec SC-ZObis where**

$$\begin{aligned}
& \text{SC-ZObis } (\text{Atm } \text{atm}) \longleftrightarrow \text{compatAtm } \text{atm} \\
| \text{SC-ZObis } (\text{Seq } c1 \ c2) & \longleftrightarrow (\text{SC-siso } c1 \wedge \text{SC-ZObis } c2) \vee \\
& (\text{SC-ZObis } c1 \wedge \text{SC-discr } c2) \vee \\
& \text{SC-discr } (\text{Seq } c1 \ c2) \vee \\
& \text{SC-siso } (\text{Seq } c1 \ c2) \\
| \text{SC-ZObis } (\text{If } \text{tst } c1 \ c2) & \longleftrightarrow (\text{if compatTst } \text{tst} \\
& \text{then } (\text{SC-ZObis } c1 \wedge \text{SC-ZObis } c2) \\
& \text{else } (\text{SC-discr } (\text{If } \text{tst } c1 \ c2) \vee \\
& \text{SC-siso } (\text{If } \text{tst } c1 \ c2))) \\
| \text{SC-ZObis } (\text{While } \text{tst } c) & \longleftrightarrow \text{SC-discr } (\text{While } \text{tst } c) \vee \\
& \text{SC-siso } (\text{While } \text{tst } c) \\
| \text{SC-ZObis } (\text{Par } c1 \ c2) & \longleftrightarrow (\text{SC-ZObis } c1 \wedge \text{SC-ZObis } c2) \vee \\
& \text{SC-discr } (\text{Par } c1 \ c2) \vee \\
& \text{SC-siso } (\text{Par } c1 \ c2)
\end{aligned}$$

**primrec SC-Wbis where**

$$\begin{aligned}
& \text{SC-Wbis } (\text{Atm } \text{atm}) \longleftrightarrow \text{compatAtm } \text{atm} \\
| \text{SC-Wbis } (\text{Seq } c1 \ c2) & \longleftrightarrow (\text{SC-WbisT } c1 \wedge \text{SC-Wbis } c2) \vee \\
& (\text{SC-Wbis } c1 \wedge \text{SC-discr } c2) \vee \\
& \text{SC-ZObis } (\text{Seq } c1 \ c2) \vee \\
& \text{SC-WbisT } (\text{Seq } c1 \ c2) \\
| \text{SC-Wbis } (\text{If } \text{tst } c1 \ c2) & \longleftrightarrow (\text{if compatTst } \text{tst} \\
& \text{then } (\text{SC-Wbis } c1 \wedge \text{SC-Wbis } c2) \\
& \text{else } (\text{SC-ZObis } (\text{If } \text{tst } c1 \ c2) \vee \\
& \text{SC-WbisT } (\text{If } \text{tst } c1 \ c2))) \\
| \text{SC-Wbis } (\text{While } \text{tst } c) & \longleftrightarrow \text{SC-ZObis } (\text{While } \text{tst } c) \vee \\
& \text{SC-WbisT } (\text{While } \text{tst } c) \\
| \text{SC-Wbis } (\text{Par } c1 \ c2) & \longleftrightarrow (\text{SC-Wbis } c1 \wedge \text{SC-Wbis } c2) \vee \\
& \text{SC-ZObis } (\text{Par } c1 \ c2) \vee \\
& \text{SC-WbisT } (\text{Par } c1 \ c2)
\end{aligned}$$

**primrec SC-BisT where**

$$\text{SC-BisT } (\text{Atm } \text{atm}) \longleftrightarrow \text{compatAtm } \text{atm}$$

$| \text{SC-BisT } (\text{Seq } c1 \ c2) \longleftrightarrow (\text{SC-BisT } c1 \wedge \text{SC-BisT } c2) \vee$   
 $\text{SC-discr } (\text{Seq } c1 \ c2) \vee$   
 $\text{SC-WbisT } (\text{Seq } c1 \ c2)$   
 $| \text{SC-BisT } (\text{If } \text{tst } c1 \ c2) \longleftrightarrow (\text{if } \text{compatTst } \text{tst}$   
 $\text{then } (\text{SC-BisT } c1 \wedge \text{SC-BisT } c2)$   
 $\text{else } (\text{SC-discr } (\text{If } \text{tst } c1 \ c2) \vee$   
 $\text{SC-WbisT } (\text{If } \text{tst } c1 \ c2)))$   
 $| \text{SC-BisT } (\text{While } \text{tst } c) \longleftrightarrow (\text{if } \text{compatTst } \text{tst}$   
 $\text{then } \text{SC-BisT } c$   
 $\text{else } (\text{SC-discr } (\text{While } \text{tst } c) \vee$   
 $\text{SC-WbisT } (\text{While } \text{tst } c)))$   
 $| \text{SC-BisT } (\text{Par } c1 \ c2) \longleftrightarrow (\text{SC-BisT } c1 \wedge \text{SC-BisT } c2) \vee$   
 $\text{SC-discr } (\text{Par } c1 \ c2) \vee$   
 $\text{SC-WbisT } (\text{Par } c1 \ c2)$

**theorem** *SC-discr[intro]*:  $\text{SC-discr } c \implies \text{discr } c$   
*<proof>*

**theorem** *SC-iso[intro]*:  $\text{SC-iso } c \implies \text{iso } c$   
*<proof>*

**theorem** *SC-iso-imp-SC-WbisT[intro]*:  $\text{SC-iso } c \implies \text{SC-WbisT } c$   
*<proof>*

**theorem** *SC-discr-imp-SC-WbisT[intro]*:  $\text{no While } c \implies \text{SC-discr } c \implies \text{SC-WbisT } c$   
*<proof>*

**theorem** *SC-WbisT[intro]*:  $\text{SC-WbisT } c \implies c \approx_{wT} c$   
*<proof>*

**theorem** *SC-discr-imp-SC-ZObis[intro]*:  $\text{SC-discr } c \implies \text{SC-ZObis } c$   
*<proof>*

**theorem** *SC-iso-imp-SC-ZObis[intro]*:  $\text{SC-iso } c \implies \text{SC-ZObis } c$   
*<proof>*

**theorem** *SC-ZObis[intro]*:  $\text{SC-ZObis } c \implies c \approx_{01} c$   
*<proof>*

**theorem** *SC-ZObis-imp-SC-Wbis[intro]*:  $\text{SC-ZObis } c \implies \text{SC-Wbis } c$   
*<proof>*

**theorem** *SC-WbisT-imp-SC-Wbis[intro]*:  $\text{SC-WbisT } c \implies \text{SC-Wbis } c$   
*<proof>*

**theorem** *SC-Wbis[intro]*:  $\text{SC-Wbis } c \implies c \approx_w c$   
*<proof>*

**theorem** *SC-discr-imp-SC-BisT[intro]*:  $SC-discr\ c \implies SC-BisT\ c$   
*<proof>*

**theorem** *SC-WbisT-imp-SC-BisT[intro]*:  $SC-WbisT\ c \implies SC-BisT\ c$   
*<proof>*

**theorem** *SC-ZObis-imp-SC-BisT[intro]*:  $SC-ZObis\ c \implies SC-BisT\ c$   
*<proof>*

**theorem** *SC-Wbis-imp-SC-BisT[intro]*:  $SC-Wbis\ c \implies SC-BisT\ c$   
*<proof>*

**theorem** *SC-BisT[intro]*:  $SC-BisT\ c \implies c \approx T\ c$   
*<proof>*

**theorem** *SC-WbisT-While*:  $SC-WbisT\ (While\ tst\ c) \longleftrightarrow SC-WbisT\ c \wedge compatTst\ tst$   
*<proof>*

**theorem** *SC-ZObis-While*:  $SC-ZObis\ (While\ tst\ c) \longleftrightarrow (compatTst\ tst \wedge SC-iso\ c) \vee SC-discr\ c$   
*<proof>*

**theorem** *SC-ZObis-If*:  $SC-ZObis\ (If\ tst\ c1\ c2) \longleftrightarrow (if\ compatTst\ tst\ then\ SC-ZObis\ c1 \wedge SC-ZObis\ c2\ else\ SC-discr\ c1 \wedge SC-discr\ c2)$   
*<proof>*

**theorem** *SC-WbisT-Seq*:  $SC-WbisT\ (Seq\ c1\ c2) \longleftrightarrow (SC-WbisT\ c1 \wedge SC-WbisT\ c2)$   
*<proof>*

**theorem** *SC-ZObis-Seq*:  $SC-ZObis\ (Seq\ c1\ c2) \longleftrightarrow (SC-iso\ c1 \wedge SC-ZObis\ c2) \vee (SC-ZObis\ c1 \wedge SC-discr\ c2)$   
*<proof>*

**theorem** *SC-Wbis-Seq*:  $SC-Wbis\ (Seq\ c1\ c2) \longleftrightarrow (SC-WbisT\ c1 \wedge SC-Wbis\ c2) \vee (SC-Wbis\ c1 \wedge SC-discr\ c2)$   
*<proof>*

**theorem** *SC-BisT-Par*:  
 $SC-BisT\ (Par\ c1\ c2) \longleftrightarrow (SC-BisT\ c1 \wedge SC-BisT\ c2)$   
*<proof>*

**end**

**end**

## 6 After-execution security

**theory** *After-Execution*  
**imports** *During-Execution*  
**begin**

**context** *PL-Indis*  
**begin**

### 6.1 Setup for bisimilarities

**lemma** *Sbis-transC*[*consumes 3, case-names Match*]:  
**assumes**  $0: c \approx_s d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow_c (c',s')$   
**obtains**  $d' t'$  **where**  
 $(d,t) \rightarrow_c (d',t')$  **and**  $s' \approx t'$  **and**  $c' \approx_s d'$   
*<proof>*

**lemma** *Sbis-transT*[*consumes 3, case-names Match*]:  
**assumes**  $0: c \approx_s d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow_t s'$   
**obtains**  $t'$  **where**  $(d,t) \rightarrow_t t'$  **and**  $s' \approx t'$   
*<proof>*

**lemma** *Sbis-transC2*[*consumes 3, case-names Match*]:  
**assumes**  $0: c \approx_s d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow_c (d',t')$   
**obtains**  $c' s'$  **where**  
 $(c,s) \rightarrow_c (c',s')$  **and**  $s' \approx t'$  **and**  $c' \approx_s d'$   
*<proof>*

**lemma** *Sbis-transT2*[*consumes 3, case-names Match*]:  
**assumes**  $0: c \approx_s d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow_t t'$   
**obtains**  $s'$  **where**  $(c,s) \rightarrow_t s'$  **and**  $s' \approx t'$   
*<proof>*

**lemma** *ZObisT-transC*[*consumes 3, case-names Match MatchS*]:  
**assumes**  $0: c \approx_{01T} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow_c (c',s')$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow_c (d',t'); s' \approx t'; c' \approx_{01T} d' \rrbracket \implies \textit{thesis}$   
**and**  $\llbracket s' \approx t; c' \approx_{01T} d \rrbracket \implies \textit{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *ZObisT-transT*[*consumes 3, case-names Match*]:  
**assumes**  $0: c \approx_{01T} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow_t s'$   
**obtains**  $t'$  **where**  $(d,t) \rightarrow_t t'$  **and**  $s' \approx t'$   
*<proof>*

**lemma** *ZObisT-transC2*[*consumes 3, case-names Match MatchS*]:

**assumes** 0:  $c \approx_{01T} d$  **and** 2:  $s \approx t$  **and** 3:  $(d,t) \rightarrow c (d',t')$   
**and** 4:  $\bigwedge c' s'. \llbracket (c,s) \rightarrow c (c',s'); s' \approx t'; c' \approx_{01T} d' \rrbracket \implies \textit{thesis}$   
**and** 5:  $\llbracket s \approx t'; c \approx_{01T} d' \rrbracket \implies \textit{thesis}$   
**shows** *thesis*  
 $\langle \textit{proof} \rangle$

**lemma** *ZObisT-transT2*[*consumes 3, case-names Match*]:  
**assumes** 0:  $c \approx_{01T} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow t t'$   
**obtains**  $s'$  **where**  $(c,s) \rightarrow t s'$  **and**  $s' \approx t'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-transC*[*consumes 3, case-names Match*]:  
**assumes** 0:  $c \approx_{wT} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow c (c',s')$   
**obtains**  $d' t'$  **where**  
 $(d,t) \rightarrow *c (d',t')$  **and**  $s' \approx t'$  **and**  $c' \approx_{wT} d'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-transT*[*consumes 3, case-names Match*]:  
**assumes** 0:  $c \approx_{wT} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow t s'$   
**obtains**  $t'$  **where**  $(d,t) \rightarrow *t t'$  **and**  $s' \approx t'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-transC2*[*consumes 3, case-names Match*]:  
**assumes** 0:  $c \approx_{wT} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow c (d',t')$   
**obtains**  $c' s'$  **where**  
 $(c,s) \rightarrow *c (c',s')$  **and**  $s' \approx t'$  **and**  $c' \approx_{wT} d'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-transT2*[*consumes 3, case-names Match*]:  
**assumes** 0:  $c \approx_{wT} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow t t'$   
**obtains**  $s'$  **where**  $(c,s) \rightarrow *t s'$  **and**  $s' \approx t'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-MtransC*[*consumes 3, case-names Match*]:  
**assumes** 1:  $c \approx_{wT} d$  **and** 2:  $s \approx t$  **and** 3:  $(c,s) \rightarrow *c (c',s')$   
**obtains**  $d' t'$  **where**  
 $(d,t) \rightarrow *c (d',t')$  **and**  $s' \approx t'$  **and**  $c' \approx_{wT} d'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-MtransT*[*consumes 3, case-names Match*]:  
**assumes** 1:  $c \approx_{wT} d$  **and** 2:  $s \approx t$  **and** 3:  $(c,s) \rightarrow *t s'$   
**obtains**  $t'$  **where**  $(d,t) \rightarrow *t t'$  **and**  $s' \approx t'$   
 $\langle \textit{proof} \rangle$

**lemma** *WbisT-MtransC2*[*consumes 3, case-names Match*]:  
**assumes**  $c \approx_{wT} d$  **and**  $s \approx t$  **and** 1:  $(d,t) \rightarrow *c (d',t')$   
**obtains**  $c' s'$  **where**

$(c,s) \rightarrow^* c (c',s')$  **and**  $s' \approx t'$  **and**  $c' \approx_{wT} d'$   
 ⟨proof⟩

**lemma** *WbisT-MtransT2*[consumes 3, case-names Match]:  
**assumes**  $c \approx_{wT} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow^* t'$   
**obtains**  $s'$  **where**  $(c,s) \rightarrow^* t s'$  **and**  $s' \approx t'$   
 ⟨proof⟩

**lemma** *ZObis-transC*[consumes 3, case-names Match MatchO MatchS]:  
**assumes**  $0: c \approx_{01} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow c (c',s')$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow c (d',t'); s' \approx t'; c' \approx_{01} d' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge t'. \llbracket (d,t) \rightarrow t t'; s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**and**  $\llbracket s' \approx t; c' \approx_{01} d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
 ⟨proof⟩

**lemma** *ZObis-transT*[consumes 3, case-names Match MatchO MatchS]:  
**assumes**  $0: c \approx_{01} d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow t s'$   
**and**  $\bigwedge t'. \llbracket (d,t) \rightarrow t t'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow c (d',t'); s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**and**  $\llbracket s' \approx t; \text{discr } d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
 ⟨proof⟩

**lemma** *ZObis-transC2*[consumes 3, case-names Match MatchO MatchS]:  
**assumes**  $0: c \approx_{01} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow c (d',t')$   
**and**  $\bigwedge c' s'. \llbracket (c,s) \rightarrow c (c',s'); s' \approx t'; c' \approx_{01} d' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge s'. \llbracket (c,s) \rightarrow t s'; s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**and**  $\llbracket s \approx t'; c \approx_{01} d' \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
 ⟨proof⟩

**lemma** *ZObis-transT2*[consumes 3, case-names Match MatchO MatchS]:  
**assumes**  $0: c \approx_{01} d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow t t'$   
**and**  $\bigwedge s'. \llbracket (c,s) \rightarrow t s'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge c' s'. \llbracket (c,s) \rightarrow c (c',s'); s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**and**  $\llbracket s \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
 ⟨proof⟩

**lemma** *Wbis-transC*[consumes 3, case-names Match MatchO]:  
**assumes**  $0: c \approx_w d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow c (c',s')$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow^* c (d',t'); s' \approx t'; c' \approx_w d' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge t'. \llbracket (d,t) \rightarrow^* t t'; s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
 ⟨proof⟩

**lemma** *Wbis-transT*[consumes 3, case-names Match MatchO]:  
**assumes**  $0: c \approx_w d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow t s'$   
**and**  $\bigwedge t'. \llbracket (d,t) \rightarrow^* t t'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow^* c (d',t'); s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-transC2*[consumes 3, case-names Match MatchO]:  
**assumes**  $0: c \approx_w d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow c (d',t')$   
**and**  $\bigwedge c' s'. \llbracket (c,s) \rightarrow^* c (c',s'); s' \approx t'; c' \approx_w d \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge s'. \llbracket (c,s) \rightarrow^* t s'; s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-transT2*[consumes 3, case-names Match MatchO]:  
**assumes**  $0: c \approx_w d$  **and**  $s \approx t$  **and**  $(d,t) \rightarrow t t'$   
**and**  $\bigwedge s'. \llbracket (c,s) \rightarrow^* t s'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge c' s'. \llbracket (c,s) \rightarrow^* c (c',s'); s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-MtransC*[consumes 3, case-names Match MatchO]:  
**assumes**  $c \approx_w d$  **and**  $s \approx t$  **and**  $(c,s) \rightarrow^* c (c',s')$   
**and**  $\bigwedge d' t'. \llbracket (d,t) \rightarrow^* c (d',t'); s' \approx t'; c' \approx_w d \rrbracket \implies \text{thesis}$   
**and**  $\bigwedge t'. \llbracket (d,t) \rightarrow^* t t'; s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-MtransT*[consumes 3, case-names Match MatchO]:  
**assumes**  $c-d: c \approx_w d$  **and**  $st: s \approx t$  **and**  $cs: (c,s) \rightarrow^* t s'$   
**and**  $1: \bigwedge t'. \llbracket (d,t) \rightarrow^* t t'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $2: \bigwedge d' t'. \llbracket (d,t) \rightarrow^* c (d',t'); s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-MtransC2*[consumes 3, case-names Match MatchO]:  
**assumes**  $c \approx_w d$  **and**  $s \approx t$  **and**  $dt: (d,t) \rightarrow^* c (d',t')$   
**and**  $1: \bigwedge c' s'. \llbracket (c,s) \rightarrow^* c (c',s'); s' \approx t'; c' \approx_w d \rrbracket \implies \text{thesis}$   
**and**  $2: \bigwedge s'. \llbracket (c,s) \rightarrow^* t s'; s' \approx t'; \text{discr } d \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*

**lemma** *Wbis-MtransT2*[consumes 3, case-names Match MatchO]:  
**assumes**  $c \approx_w d$  **and**  $s \approx t$  **and**  $dt: (d,t) \rightarrow^* t t'$   
**and**  $1: \bigwedge s'. \llbracket (c,s) \rightarrow^* t s'; s' \approx t' \rrbracket \implies \text{thesis}$   
**and**  $2: \bigwedge c' s'. \llbracket (c,s) \rightarrow^* c (c',s'); s' \approx t'; \text{discr } c \rrbracket \implies \text{thesis}$   
**shows** *thesis*  
*<proof>*



**lemma** *BisT-transC*[consumes 5, case-names Match]:  
**assumes**  $0: c \approx T d$   
**and**  $mustT\ c\ s$  **and**  $mustT\ d\ t$   
**and**  $s \approx t$  **and**  $(c, s) \rightarrow c\ (c', s')$   
**obtains**  $d'\ t'$  **where**  
 $(d, t) \rightarrow *c\ (d', t')$  **and**  $s' \approx t'$  **and**  $c' \approx T d'$   
 $\langle proof \rangle$

**lemma** *BisT-transT*[consumes 5, case-names Match]:  
**assumes**  $0: c \approx T d$   
**and**  $mustT\ c\ s$  **and**  $mustT\ d\ t$   
**and**  $s \approx t$  **and**  $(c, s) \rightarrow t\ s'$   
**obtains**  $t'$  **where**  $(d, t) \rightarrow *t\ t'$  **and**  $s' \approx t'$   
 $\langle proof \rangle$

**lemma** *BisT-transC2*[consumes 5, case-names Match]:  
**assumes**  $0: c \approx T d$   
**and**  $mustT\ c\ s$  **and**  $mustT\ d\ t$   
**and**  $s \approx t$  **and**  $(d, t) \rightarrow c\ (d', t')$   
**obtains**  $c'\ s'$  **where**  
 $(c, s) \rightarrow *c\ (c', s')$  **and**  $s' \approx t'$  **and**  $c' \approx T d'$   
 $\langle proof \rangle$

**lemma** *BisT-transT2*[consumes 5, case-names Match]:  
**assumes**  $0: c \approx T d$   
**and**  $mustT\ c\ s$  **and**  $mustT\ d\ t$   
**and**  $s \approx t$  **and**  $(d, t) \rightarrow t\ t'$   
**obtains**  $s'$  **where**  $(c, s) \rightarrow *t\ s'$  **and**  $s' \approx t'$   
 $\langle proof \rangle$

**lemma** *BisT-MtransC*[consumes 5, case-names Match]:  
**assumes**  $c \approx T d$   
**and**  $mustT\ c\ s$   $mustT\ d\ t$   
**and**  $s \approx t$  **and**  $(c, s) \rightarrow *c\ (c', s')$   
**obtains**  $d'\ t'$  **where**  
 $(d, t) \rightarrow *c\ (d', t')$  **and**  $s' \approx t'$  **and**  $c' \approx T d'$   
 $\langle proof \rangle$

**lemma** *BisT-MtransT*[consumes 5, case-names Match]:  
**assumes**  $1: c \approx T d$   
**and**  $ter: mustT\ c\ s\ mustT\ d\ t$   
**and**  $2: s \approx t$  **and**  $3: (c, s) \rightarrow *t\ s'$   
**obtains**  $t'$  **where**  $(d, t) \rightarrow *t\ t'$  **and**  $s' \approx t'$   
 $\langle proof \rangle$

**lemma** *BisT-MtransC2*[consumes 3, case-names Match]:

**assumes**  $c \approx T d$   
**and**  $ter: mustT c s mustT d t$   
**and**  $s \approx t$  **and**  $1: (d, t) \rightarrow^* c (d', t')$   
**obtains**  $c' s'$  **where**  
 $(c, s) \rightarrow^* c (c', s')$  **and**  $s' \approx t'$  **and**  $c' \approx T d'$   
 $\langle proof \rangle$

**lemma** *BisT-MtransT02*[*consumes 3, case-names Match*]:  
**assumes**  $c \approx T d$   
**and**  $ter: mustT c s mustT d t$   
**and**  $s \approx t$  **and**  $(d, t) \rightarrow^* t'$   
**obtains**  $s'$  **where**  $(c, s) \rightarrow^* t s'$  **and**  $s' \approx t'$   
 $\langle proof \rangle$

## 6.2 Execution traces

**primrec** *parTrace* **where**  
 $parTrace \ [] \longleftrightarrow False$  |  
 $parTrace (cf \# cfl) \longleftrightarrow (cfl \neq [] \longrightarrow parTrace cfl \wedge cf \rightarrow c hd cfl)$

**lemma** *trans-Step2*:  
 $cf \rightarrow^* c cf' \Longrightarrow cf' \rightarrow c cf'' \Longrightarrow cf \rightarrow^* c cf''$   
 $\langle proof \rangle$

**lemma** *parTrace-not-empty*[*simp*]:  $parTrace cfl \Longrightarrow cfl \neq []$   
 $\langle proof \rangle$

**lemma** *parTrace-snoc*[*simp*]:  
 $parTrace (cfl @ [cf]) \longleftrightarrow (cfl \neq [] \longrightarrow parTrace cfl \wedge last cfl \rightarrow c cf)$   
 $\langle proof \rangle$

**lemma** *MtransC-Ex-parTrace*:  
**assumes**  $cf \rightarrow^* c cf'$  **shows**  $\exists cfl. parTrace cfl \wedge hd cfl = cf \wedge last cfl = cf'$   
 $\langle proof \rangle$

**lemma** *parTrace-imp-MtransC*:  
**assumes**  $pT: parTrace cfl$   
**shows**  $(hd cfl) \rightarrow^* c (last cfl)$   
 $\langle proof \rangle$

**fun** *finTrace* **where**  
 $finTrace (cfl, s) \longleftrightarrow$   
 $parTrace cfl \wedge last cfl \rightarrow t s$

**declare** *finTrace.simps*[*simp del*]

**definition** *lengthFT*  $tr \equiv Suc (length (fst tr))$

**definition**  $fstate\ tr \equiv snd\ tr$

**definition**  $iconfig\ tr \equiv hd\ (fst\ tr)$

**lemma** *MtransT-Ex-finTrace*:

**assumes**  $cf \rightarrow *t\ s$  **shows**  $\exists tr. finTrace\ tr \wedge iconfig\ tr = cf \wedge fstate\ tr = s$   
*<proof>*

**lemma** *finTrace-imp-MtransT*:

$finTrace\ tr \implies iconfig\ tr \rightarrow *t\ fstate\ tr$   
*<proof>*

### 6.3 Relationship between during-execution and after-execution security

**lemma** *WbisT-trace2*:

**assumes**  $bis: c \approx wT\ d\ s \approx t$   
**and**  $tr: finTrace\ tr\ iconfig\ tr = (c, s)$   
**shows**  $\exists tr'. finTrace\ tr' \wedge iconfig\ tr' = (d, t) \wedge fstate\ tr \approx fstate\ tr'$   
*<proof>*

**theorem** *WbisT-trace*:

**assumes**  $c \approx wT\ c$  **and**  $s \approx t$   
**and**  $finTrace\ tr$  **and**  $iconfig\ tr = (c, s)$   
**shows**  $\exists tr'. finTrace\ tr' \wedge iconfig\ tr' = (c, t) \wedge fstate\ tr \approx fstate\ tr'$   
*<proof>*

**theorem** *ZObisT-trace2*:

**assumes**  $bis: c \approx 01T\ d\ s \approx t$   
**and**  $tr: finTrace\ tr\ iconfig\ tr = (c, s)$   
**shows**  $\exists tr'. finTrace\ tr' \wedge iconfig\ tr' = (d, t) \wedge$   
 $fstate\ tr \approx fstate\ tr' \wedge lengthFT\ tr' \leq lengthFT\ tr$   
*<proof>*

**theorem** *ZObisT-trace*:

**assumes**  $c \approx 01T\ c$  **and**  $s \approx t$   
**and**  $finTrace\ tr\ iconfig\ tr = (c, s)$   
**shows**  $\exists tr'. finTrace\ tr' \wedge iconfig\ tr' = (c, t) \wedge$   
 $fstate\ tr \approx fstate\ tr' \wedge lengthFT\ tr' \leq lengthFT\ tr$   
*<proof>*

**theorem** *Sbis-trace*:

**assumes**  $bis: c \approx s\ d\ s \approx t$   
**and**  $tr: finTrace\ tr\ iconfig\ tr = (c, s)$   
**shows**  $\exists tr'. finTrace\ tr' \wedge iconfig\ tr' = (d, t) \wedge fstate\ tr \approx fstate\ tr' \wedge$   
 $lengthFT\ tr' = lengthFT\ tr$

*<proof>*

**corollary** *siso-trace*:

**assumes** *siso*  $c$  **and**  $s \approx t$

**and** *finTrace*  $tr$  **and** *iconfig*  $tr = (c,s)$

**shows**

$\exists tr'. \text{finTrace } tr' \wedge \text{iconfig } tr' = (c,t) \wedge \text{fstate } tr \approx \text{fstate } tr'$   
 $\wedge \text{lengthFT } tr' = \text{lengthFT } tr$

*<proof>*

**theorem** *Wbis-trace*:

**assumes**  $T: \bigwedge s. \text{mustT } c s$

**and** *bis*:  $c \approx_w c s \approx t$

**and**  $tr: \text{finTrace } tr \text{ iconfig } tr = (c,s)$

**shows**  $\exists tr'. \text{finTrace } tr' \wedge \text{iconfig } tr' = (c,t) \wedge \text{fstate } tr \approx \text{fstate } tr'$

*<proof>*

**corollary** *ZObis-trace*:

**assumes**  $T: \bigwedge s. \text{mustT } c s$

**and** *ZObis*:  $c \approx_{01} c$  **and** *indis*:  $s \approx t$

**and**  $tr: \text{finTrace } tr \text{ iconfig } tr = (c,s)$

**shows**  $\exists tr'. \text{finTrace } tr' \wedge \text{iconfig } tr' = (c,t) \wedge \text{fstate } tr \approx \text{fstate } tr'$

*<proof>*

**theorem** *BisT-trace*:

**assumes** *bis*:  $c \approx_T c s \approx t$

**and**  $T: \text{mustT } c s \text{ mustT } c t$

**and**  $tr: \text{finTrace } tr \text{ iconfig } tr = (c,s)$

**shows**  $\exists tr'. \text{finTrace } tr' \wedge \text{iconfig } tr' = (c,t) \wedge \text{fstate } tr \approx \text{fstate } tr'$

*<proof>*

**end**

**end**

## 7 Concrete setting

**theory** *Concrete*

**imports** *Syntactic-Criteria After-Execution*

**begin**

**lemma** (in *PL-Indis*) *WbisT-If-cross*:

**assumes**  $c1 \approx_w T c2 \ c1 \approx_w T c1 \ c2 \approx_w T c2$

**shows**  $(\text{If } \text{tst } c1 \ c2) \approx_w T (\text{If } \text{tst } c1 \ c2)$

*<proof>*

We instantiate the following notions, kept generic so far:

- On the language syntax:
  - atoms, tests and states just like at the possibilistic case;
  - choices, to either if-choices (based on tests) or binary fixed-probability choices;
  - the schedulers, to the uniform one
- On the security semantics, the lattice of levels and the indis relation, again, just like at the possibilistic case.

**datatype** *level* = *Lo* | *Hi*

**lemma** [*simp*]:  $\bigwedge l. l \neq Hi \longleftrightarrow l = Lo$  **and**

[*simp*]:  $\bigwedge l. Hi \neq l \longleftrightarrow Lo = l$  **and**

[*simp*]:  $\bigwedge l. l \neq Lo \longleftrightarrow l = Hi$  **and**

[*simp*]:  $\bigwedge l. Lo \neq l \longleftrightarrow Hi = l$

*<proof>*

**lemma** [*dest*]:  $\bigwedge l A. [l \in A; Lo \notin A] \Longrightarrow l = Hi$  **and**

[*dest*]:  $\bigwedge l A. [l \in A; Hi \notin A] \Longrightarrow l = Lo$

*<proof>*

**declare** *level.split*[*split*]

**instantiation** *level* :: *complete-lattice*

**begin**

**definition** *top-level*: *top*  $\equiv Hi$

**definition** *bot-level*: *bot*  $\equiv Lo$

**definition** *inf-level*: *inf* *l1* *l2*  $\equiv$  if *Lo*  $\in$  {*l1*,*l2*} then *Lo* else *Hi*

**definition** *sup-level*: *sup* *l1* *l2*  $\equiv$  if *Hi*  $\in$  {*l1*,*l2*} then *Hi* else *Lo*

**definition** *less-eq-level*: *less-eq* *l1* *l2*  $\equiv$  (*l1* = *Lo*  $\vee$  *l2* = *Hi*)

**definition** *less-level*: *less* *l1* *l2*  $\equiv$  *l1* = *Lo*  $\wedge$  *l2* = *Hi*

**definition** *Inf-level*: *Inf* *L*  $\equiv$  if *Lo*  $\in$  *L* then *Lo* else *Hi*

**definition** *Sup-level*: *Sup* *L*  $\equiv$  if *Hi*  $\in$  *L* then *Hi* else *Lo*

**instance**

*<proof>*

**end**

**lemma** *sup-eq-Lo*[*simp*]: *sup* *a* *b* = *Lo*  $\longleftrightarrow$  *a* = *Lo*  $\wedge$  *b* = *Lo*

*<proof>*

**datatype** *var* = *h* | *h'* | *l* | *l'*

**datatype** *exp* = *Ct nat* | *Var var* | *Plus exp exp* | *Minus exp exp*

**datatype** *test* = *Tr* | *Eq exp exp* | *Gt exp exp* | *Non test*

**datatype** *atom* = *Assign var exp*  
**type-synonym** *state* = *var*  $\Rightarrow$  *nat*

**syntax**  
*-assign* :: '*a*  $\Rightarrow$  '*a*  $\Rightarrow$  '*a* (- ::= - [1000, 61] 61)

**translations**  
*x* ::= *expr* == *CONST Atm (CONST Assign x expr)*

**primrec** *sec* **where**  
*sec h* = *Hi*  
| *sec h'* = *Hi*  
| *sec l* = *Lo*  
| *sec l'* = *Lo*

**fun** *eval* **where**  
*eval (Ct n) s* = *n*  
| *eval (Var x) s* = *s x*  
| *eval (Plus e1 e2) s* = *eval e1 s + eval e2 s*  
| *eval (Minus e1 e2) s* = *eval e1 s - eval e2 s*

**fun** *tval* **where**  
*tval Tr s* = *True*  
| *tval (Eq e1 e2) s* = (*eval e1 s = eval e2 s*)  
| *tval (Gt e1 e2) s* = (*eval e1 s > eval e2 s*)  
| *tval (Non e) s* = ( $\neg$  *tval e s*)

**fun** *aval* **where**  
*aval (Assign x e) s* = (*s (x := eval e s)*)

**definition** *indis* :: (*state \* state*) **setwhere**  
*indis*  $\equiv$   $\{(s,t). \text{ALL } x. \text{sec } x = \text{Lo} \longrightarrow s \ x = t \ x\}$

**interpretation** *Example-PL: PL-Indis tval aval indis*  
<*proof*>

**fun** *exprSec* **where**  
*exprSec (Ct n)* = *bot*  
| *exprSec (Var x)* = *sec x*  
| *exprSec (Plus e1 e2)* = *sup (exprSec e1) (exprSec e2)*  
| *exprSec (Minus e1 e2)* = *sup (exprSec e1) (exprSec e2)*

**fun** *tstSec* **where**  
*tstSec Tr* = *bot*  
| *tstSec (Eq e1 e2)* = *sup (exprSec e1) (exprSec e2)*  
| *tstSec (Gt e1 e2)* = *sup (exprSec e1) (exprSec e2)*  
| *tstSec (Non e)* = *tstSec e*

**lemma** *exprSec-Lo-eval-eq*: *exprSec expr = Lo*  $\implies$  (*s, t*)  $\in$  *indis*  $\implies$  *eval expr s*

= *eval expr t*  
(*proof*)

**lemma** *compatAtmSyntactic[simp]*: *exprSec expr = Lo*  $\vee$  *sec v = Hi*  $\implies$  *Example-PL.compatAtm (Assign v expr)*  
(*proof*)

**lemma** *presAtmSyntactic[simp]*: *sec v = Hi*  $\implies$  *Example-PL.presAtm (Assign v expr)*  
(*proof*)

**lemma** *compatTstSyntactic[simp]*: *tstSec tst = Lo*  $\implies$  *Example-PL.compatTst tst*  
(*proof*)

**lemma** *Example-PL.SC-discr (h ::= Ct 0)*  
(*proof*)

**abbreviation** *siso c*  $\equiv$  *Example-PL.siso c*

**abbreviation** *siso0 c*  $\equiv$  *Example-PL.siso0 c*

**abbreviation** *discr c*  $\equiv$  *Example-PL.discr c*

**abbreviation** *discr0 c*  $\equiv$  *Example-PL.discr0 c*

**abbreviation** *Sbis-abbrev (infix  $\approx_s$  55)* **where** *c1  $\approx_s$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.Sbis*

**abbreviation** *ZObis-abbrev (infix  $\approx_{01}$  55)* **where** *c1  $\approx_{01}$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.ZObis*

**abbreviation** *ZObisT-abbrev (infix  $\approx_{01T}$  55)* **where** *c1  $\approx_{01T}$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.ZObisT*

**abbreviation** *Wbis-abbrev (infix  $\approx_w$  55)* **where** *c1  $\approx_w$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.Wbis*

**abbreviation** *WbisT-abbrev (infix  $\approx_{wT}$  55)* **where** *c1  $\approx_{wT}$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.WbisT*

**abbreviation** *BisT-abbrev (infix  $\approx_T$  55)* **where** *c1  $\approx_T$  c2*  $\equiv$  *(c1,c2)  $\in$  Example-PL.BisT*

## 7.1 Programs from EXAMPLE 1

**definition** [*simp*]: *c0 = (h ::= Ct 0)*

**definition** [*simp*]: *c1 = (if Eq (Var l) (Ct 0) then h ::= Ct 1 else l ::= Ct 2)*

**definition** [*simp*]: *c2 = (if Eq (Var h) (Ct 0) then h ::= Ct 1 else h ::= Ct 2)*

**definition** [*simp*]: *c3 = (if Eq (Var h) (Ct 0) then h ::= Ct 1 ;; h ::= Ct 2  
else h ::= Ct 3)*

**definition** [*simp*]: *c4 = l ::= Ct 4 ;; c3*

**definition** [*simp*]: *c5 = c3 ;; l ::= Ct 4*

**definition** [simp]:  $c6 = l ::= \text{Var } h$

**definition** [simp]:  $c7 = l ::= \text{Var } h ;; l ::= \text{Ct } 0$

**definition** [simp]:  $c8 = h' ::= \text{Var } h ;;$   
 $\text{while } Gt (\text{Var } h) (\text{Ct } 0) \text{ do } (h ::= \text{Minus } (\text{Var } h) (\text{Ct } 1) ;; h' ::= \text{Plus } (\text{Var } h')$   
 $(\text{Ct } 1)) ;;$   
 $l ::= \text{Ct } 4$

**definition** [simp]:  $c9 = c7 \mid l' ::= \text{Var } l$

**definition** [simp]:  $c10 = c5 \mid l ::= \text{Ct } 5$

**definition** [simp]:  $c11 = c8 \mid l ::= \text{Ct } 5$

**declare** *bot-level*[iff]

**theorem** *c0*: *siso c0 discr c0*  
*<proof>*

**theorem** *c1*: *siso c1 c1  $\approx_s$  c1*  
*<proof>*

**theorem** *c2*: *discr c2*  
*<proof>*

**theorem** *Sbis-c2*: *c2  $\approx_s$  c2*  
*<proof>*

**theorem** *c3*: *discr c3*  
*<proof>*

**theorem** *c4*: *c4  $\approx_{01}$  c4*  
*<proof>*

**theorem** *c5*: *c5  $\approx_w$  c5*  
*<proof>*

Example 4 from the paper

**theorem** *c3  $\approx_{wT}$  c3* *<proof>*

**theorem** *c5  $\approx_{wT}$  c5* *<proof>*

**corollary** *discr (while Eq (Var h) (Ct 0) do h ::= Ct 0)*  
*<proof>*

Example 5 from the paper

**definition** [simp]:  $c12 \equiv h ::= \text{Ct } 4 ;;$   
 $\text{while } Gt (\text{Var } h) (\text{Ct } 0)$



$do (h ::= Minus (Var h) (Ct 1) ;; h' ::= Plus (Var h') (Ct 1)) ;;$   
 $l ::= Ct 1$

**corollary**  $(c12 \mid l ::= Ct 2) \approx T (c12 \mid l ::= Ct 2)$   
 $\langle proof \rangle$

**definition**  $[simp]$ :  $c13 =$   
 $(if Eq (Var h) (Ct 0) then h ::= Ct 1 ;; l ::= Ct 2 else l ::= Ct 2) ;; l' ::= Ct 4$

**lemma**  $c13$ -inner:  
 $(h ::= Ct 1 ;; l ::= Ct 2) \approx wT (l ::= Ct 2)$   
 $\langle proof \rangle$

**theorem**  $c13 \approx wT c13$   
 $\langle proof \rangle$

**end**

## References

- [1] A. Popescu, J. Hölzl, and T. Nipkow. Proving possibilistic, probabilistic noninterference. In *Certified Programs and Proofs (CPP) '12*, 2012.