# Probabilistic Noninterference

Andrei Popescu Johannes Hölzl

March 17, 2025

**Abstract**

We formalize a probabilistic noninterference for a multi-threaded language with uniform scheduling, where probabilistic behaviour comes from both the scheduler and the individual threads. We define notions probabilistic noninterference in two variants: resumption-based and trace-based. For the resumption-based notions, we prove compositionality w.r.t. the language constructs and establish sound type-system-like syntactic criteria.

This is a formalization of the mathematical development presented in the papers [1, 2]. It is the probabilistic variant of the Possibilistic Noninterference AFP entry.

## Contents

# 1 Simple While Language with probabilistic choice and parallel execution

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

## 1.1 Preliminaries

Trivia

**declare** *zero-le-mult-iff*[*simp*]
**declare** *split-mult-pos-le*[*simp*]
**declare** *zero-le-divide-iff*[*simp*]

**lemma** *in-minus-Un*[*simp*]:
**assumes** $i \in I$
**shows** $I - \{i\}$ *Un* $\{i\} = I$ **and** $\{i\}$ *Un* $(I - \{i\}) = I$
⟨*proof*⟩

**lemma** *less-plus-cases*[*case-names Left Right*]:
**assumes**
$*$: $(i{::}nat) < n1 \Longrightarrow phi$ **and**
$**$: $\bigwedge i2.\ i = n1 + i2 \Longrightarrow phi$
**shows** *phi*
⟨*proof*⟩

**lemma** *less-plus-elim*[*elim!*, *consumes 1*, *case-names Left Right*]:
**assumes** *i*: $(i{::}\ nat) < n1 + n2$ **and**
$*$: $i < n1 \Longrightarrow phi$ **and**
$**$: $\bigwedge i2.\ [\![i2 < n2;\ i = n1 + i2]\!] \Longrightarrow phi$
**shows** *phi*
⟨*proof*⟩

**lemma** *nth-append-singl*[*simp*]:
$i < length\ al \Longrightarrow (al\ @\ [a])\ !\ i = al\ !\ i$
⟨*proof*⟩

**lemma** *take-append-singl*[*simp*]:
**assumes** $n < length\ al$ **shows** $take\ n\ (al\ @\ [a]) = take\ n\ al$
⟨*proof*⟩

**lemma** *length-unique-prefix*:
$al1 \leq al \Longrightarrow al2 \leq al \Longrightarrow length\ al1 = length\ al2 \Longrightarrow al1 = al2$
⟨*proof*⟩

take:

**lemma** *take-length*[*simp*]:
$take\ (length\ al)\ al = al$
⟨*proof*⟩

**lemma** *take-le*:
**assumes** $n < length\ al$
**shows** $take\ n\ al\ @\ [al\ !\ n] \leq al$
⟨*proof*⟩

**lemma** *list-less-iff-prefix*: $a < b \longleftrightarrow strict\text{-}prefix\ a\ b$
⟨*proof*⟩

**lemma** *take-lt*:
$n < length\ al \Longrightarrow take\ n\ al < al$
⟨*proof*⟩

**lemma** *le-take*:
**assumes** $n1 \leq n2$
**shows** $take\ n1\ al \leq take\ n2\ al$
⟨*proof*⟩

**lemma** *inj-take*:
**assumes** $n1 \leq length\ al$ **and** $n2 \leq length\ al$
**shows** $take\ n1\ al = take\ n2\ al \longleftrightarrow n1 = n2$
⟨*proof*⟩

**lemma** *lt-take*: $n1 < n2 \Longrightarrow n2 \leq length\ al \Longrightarrow take\ n1\ al < take\ n2\ al$
⟨*proof*⟩

lsum:

**definition** *lsum* :: $('a \Rightarrow nat) \Rightarrow 'a\ list \Rightarrow nat$ **where**
$lsum\ f\ al \equiv sum\text{-}list\ (map\ f\ al)$

**lemma** *lsum-simps*[*simp*]:
$lsum\ f\ [] = 0$

*lsum f (al @ [a]) = lsum f al + f a*
⟨*proof*⟩

**lemma** *lsum-append*:
*lsum f (al @ bl) = lsum f al + lsum f bl*
⟨*proof*⟩

**lemma** *lsum-cong*[*fundef-cong*]:
**assumes** $\bigwedge$ *a. a* ∈ *set al* ⟹ *f a = g a*
**shows** *lsum f al = lsum g al*
⟨*proof*⟩

**lemma** *lsum-gt-0*[*simp*]:
**assumes** *al* ≠ [] **and** $\bigwedge$ *a. a* ∈ *set al* ⟹ *0 < f a*
**shows** *0 < lsum f al*
⟨*proof*⟩

**lemma** *lsum-mono*[*simp*]:
**assumes** *al* ≤ *bl*
**shows** *lsum f al* ≤ *lsum f bl*
⟨*proof*⟩

**lemma** *lsum-mono2*[*simp*]:
**assumes** *f*: $\bigwedge$ *b. b* ∈ *set bl* ⟹ *f b > 0* **and** *le*: *al < bl*
**shows** *lsum f al < lsum f bl*
⟨*proof*⟩

**lemma** *lsum-take*[*simp*]:
*lsum f (take n al)* ≤ *lsum f al*
⟨*proof*⟩

**lemma** *less-lsum-nchotomy*:
**assumes** *f*: $\bigwedge$ *a. a* ∈ *set al* ⟹ *0 < f a*
**and** *i*: *(i::nat) < lsum f al*
**shows** ∃ *n j. n < length al* ∧ *j < f (al ! n)* ∧ *i = lsum f (take n al) + j*
⟨*proof*⟩

**lemma** *less-lsum-unique*:
**assumes** $\bigwedge$ *a. a* ∈ *set al* ⟹ *(0::nat) < f a*
**and** *n1 < length al* ∧ *j1 < f (al ! n1)*
**and** *n2 < length al* ∧ *j2 < f (al ! n2)*
**and** *lsum f (take n1 al) + j1 = lsum f (take n2 al) + j2*
**shows** *n1 = n2* ∧ *j1 = j2*
⟨*proof*⟩

**definition** *locate-pred* **where**
*locate-pred f al (i::nat) n-j* ≡
*fst n-j < length al* ∧
*snd n-j < f (al ! (fst n-j))* ∧

$i = lsum\ f\ (take\ (fst\ n\text{-}j)\ al) + (snd\ n\text{-}j)$

**definition** *locate* **where**
$locate\ f\ al\ i \equiv SOME\ n\text{-}j.\ locate\text{-}pred\ f\ al\ i\ n\text{-}j$

**definition** *locate1* **where** $locate1\ f\ al\ i \equiv fst\ (locate\ f\ al\ i)$
**definition** *locate2* **where** $locate2\ f\ al\ i \equiv snd\ (locate\ f\ al\ i)$

**lemma** *locate-pred-ex*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow 0 < f\ a$
**and** $i < lsum\ f\ al$
**shows** $\exists\ n\text{-}j.\ locate\text{-}pred\ f\ al\ i\ n\text{-}j$
⟨*proof*⟩

**lemma** *locate-pred-unique*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow 0 < f\ a$
**and** $locate\text{-}pred\ f\ al\ i\ n1\text{-}j1\ locate\text{-}pred\ f\ al\ i\ n2\text{-}j2$
**shows** $n1\text{-}j1 = n2\text{-}j2$
⟨*proof*⟩

**lemma** *locate-locate-pred*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow (0::nat) < f\ a$
**and** $i < lsum\ f\ al$
**shows** $locate\text{-}pred\ f\ al\ i\ (locate\ f\ al\ i)$
⟨*proof*⟩

**lemma** *locate-locate-pred-unique*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow (0::nat) < f\ a$
**and** $locate\text{-}pred\ f\ al\ i\ n\text{-}j$
**shows** $n\text{-}j = locate\ f\ al\ i$
⟨*proof*⟩

**lemma** *locate*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow 0 < f\ a$
**and** $i < lsum\ f\ al$
**shows** $locate1\ f\ al\ i < length\ al\ \wedge$
$locate2\ f\ al\ i < f\ (al\ !\ (locate1\ f\ al\ i))\ \wedge$
$i = lsum\ f\ (take\ (locate1\ f\ al\ i)\ al) + (locate2\ f\ al\ i)$
⟨*proof*⟩

**lemma** *locate-unique*:
**assumes** $\bigwedge a.\ a \in set\ al \Longrightarrow 0 < f\ a$
**and** $n < length\ al$ **and** $j < f\ (al\ !\ n)$ **and** $i = lsum\ f\ (take\ n\ al) + j$
**shows** $n = locate1\ f\ al\ i \wedge j = locate2\ f\ al\ i$
⟨*proof*⟩

sum:

**lemma** *sum-2*[*simp*]:
$sum\ f\ \{..< 2\} = f\ 0 + f\ (Suc\ 0)$

⟨*proof*⟩

**lemma** *inj-Plus*[*simp*]:
*inj* ((+) (*a*::*nat*))
⟨*proof*⟩

**lemma** *inj-on-Plus*[*simp*]:
*inj-on* ((+) (*a*::*nat*)) *A*
⟨*proof*⟩

**lemma** *Plus-int*[*simp*]:
**fixes** *a* :: *nat*
**shows** (+) *b* ' {..< *a*} = {*b* ..< *b* + *a*}
⟨*proof*⟩

**lemma** *sum-minus*[*simp*]:
**fixes** *a* :: *nat*
**shows** *sum f* {*a* ..< *a* + *b*} = *sum* (%*x*. *f* (*a* + *x*)) {..< *b*}
⟨*proof*⟩

**lemma** *sum-Un-introL*:
**assumes** *A1* = *B1 Un C1* **and** *x* = *x1* + *x2*
*finite A1* **and**
*B1 Int C1* = {} **and**
*sum f1 B1* = *x1* **and** *sum f1 C1* = *x2*
**shows** *sum f1 A1* = *x*
⟨*proof*⟩

**lemma** *sum-Un-intro*:
**assumes** *A1* = *B1 Un C1* **and** *A2* = *B2 Un C2* **and**
*finite A1* **and** *finite A2* **and**
*B1 Int C1* = {} **and** *B2 Int C2* = {} **and**
*sum f1 B1* = *sum f2 B2* **and** *sum f1 C1* = *sum f2 C2*
**shows** *sum f1 A1* = *sum f2 A2*
⟨*proof*⟩

**lemma** *sum-UN-introL*:
**assumes** *A1*: *A1* = (*UN n* : *N*. *B1 n*) **and** *a2*: *a2* = *sum b2 N* **and**
*fin*: *finite N* ⋀ *n*. *n* ∈ *N* ⟹ *finite* (*B1 n*) **and**
*int*: ⋀ *m n*. {*m*, *n*} ⊆ *N* ∧ *m* ≠ *n* ⟹ *B1 m* ∩ *B1 n* = {} **and**
*ss*: ⋀ *n*. *n* ∈ *N* ⟹ *sum f1* (*B1 n*) = *b2 n*
**shows** *sum f1 A1* = *a2* (**is** *?L* = *a2*)
⟨*proof*⟩

**lemma** *sum-UN-intro*:
**assumes** *A1*: *A1* = (*UN n* : *N*. *B1 n*) **and** *A2*: *A2* = (*UN n* : *N*. *B2 n*) **and**
*fin*: *finite N* ⋀ *n*. *n* ∈ *N* ⟹ *finite* (*B1 n*) ∧ *finite* (*B2 n*) **and**
*int*: ⋀ *m n*. {*m*, *n*} ⊆ *N* ∧ *m* ≠ *n* ⟹ *B1 m* ∩ *B1 n* = {} ⋀ *m n*. {*m*, *n*} ⊆ *N*
⟹ *B2 m* ∩ *B2 n* = {} **and**

*ss*: $\bigwedge$ *n*. $n \in N \Longrightarrow$ *sum f1* (*B1 n*) = *sum f2* (*B2 n*)
**shows** *sum f1 A1* = *sum f2 A2* (**is** *?L* = *?R*)
⟨*proof*⟩

**lemma** *sum-Minus-intro*:
**fixes** *f1* :: *'a1* ⇒ *real* **and** *f2* :: *'a2* ⇒ *real*
**assumes** *B1* = *A1* − {*a1*} **and** *B2* = *A2* − {*a2*} **and**
*a1* : *A1* **and** *a2* : *A2* **and** *finite A1* **and** *finite A2*
*sum f1 A1* = *sum f2 A2* **and** *f1 a1* = *f2 a2*
**shows** *sum f1 B1* = *sum f2 B2*
⟨*proof*⟩

**lemma** *sum-singl-intro*:
**assumes** *b* = *f a*
**and** *finite A* **and** *a* ∈ *A*
**and** $\bigwedge$ *a'*. ⟦*a'* ∈ *A*; *a'* ≠ *a*⟧ ⟹ *f a'* = *0*
**shows** *sum f A* = *b*
⟨*proof*⟩

**lemma** *sum-all0-intro*:
**assumes** *b* = *0*
**and** $\bigwedge$ *a*. *a* ∈ *A* ⟹ *f a* = *0*
**shows** *sum f A* = *b*
⟨*proof*⟩

**lemma** *sum-1*:
**assumes** *I*: *finite I* **and** *ss*: *sum f I* = *1* **and** *i*: *i* ∈ *I* − *I1* **and** *I1*: *I1* ⊆ *I*
**and** *f*: ⋀*i*. *i* ∈ *I* ⟹ (*0*::*real*) ≤ *f i*
**shows** *f i* ≤ *1* − *sum f I1*
⟨*proof*⟩

## 1.2 Syntax

**datatype** (*'test*, *'atom*, *'choice*) *cmd* =
*Done*
| *Atm 'atom*
| *Seq* (*'test*, *'atom*, *'choice*) *cmd* (*'test*, *'atom*, *'choice*) *cmd* (‹- ;; -› [*60*, *61*] *60*)
| *While 'test* (*'test*, *'atom*, *'choice*) *cmd*
| *Ch 'choice* (*'test*, *'atom*, *'choice*) *cmd* (*'test*, *'atom*, *'choice*) *cmd*
| *Par* (*'test*, *'atom*, *'choice*) *cmd list*
| *ParT* (*'test*, *'atom*, *'choice*) *cmd list*

**fun** *noWhile* **where**
*noWhile Done* ⟷ *True*
| *noWhile* (*Atm atm*) ⟷ *True*
| *noWhile* (*c1* ;; *c2*) ⟷ *noWhile c1* ∧ *noWhile c2*
| *noWhile* (*While tst c*) ⟷ *False*
| *noWhile* (*Ch ch c1 c2*) ⟷ *noWhile c1* ∧ *noWhile c2*

| *noWhile* (*Par cl*) $\longleftrightarrow$ ($\forall$ $c \in$ *set cl*. *noWhile c*)
| *noWhile* (*ParT cl*) $\longleftrightarrow$ ($\forall$ $c \in$ *set cl*. *noWhile c*)

**fun** *finished* **where**
*finished Done* $\longleftrightarrow$ *True*
| *finished* (*Atm atm*) $\longleftrightarrow$ *False*
| *finished* (*c1* ;; *c2*) $\longleftrightarrow$ *False*
| *finished* (*While tst c*) $\longleftrightarrow$ *False*
| *finished* (*Ch ch c1 c2*) $\longleftrightarrow$ *False*
| *finished* (*Par cl*) $\longleftrightarrow$ ($\forall$ $c \in$ *set cl*. *finished c*)
| *finished* (*ParT cl*) $\longleftrightarrow$ ($\forall$ $c \in$ *set cl*. *finished c*)

**definition** *noWhileL* **where**
*noWhileL cl* $\equiv \forall$ $c \in$ *set cl*. *noWhile c*

**lemma** *fin-Par-noWhileL*[*simp*]:
*noWhile* (*Par cl*) $\longleftrightarrow$ *noWhileL cl*
⟨*proof*⟩

**lemma** *fin-ParT-noWhileL*[*simp*]:
*noWhile* (*ParT cl*) $\longleftrightarrow$ *noWhileL cl*
⟨*proof*⟩

**declare** *noWhile.simps*(*6*) [*simp del*]
**declare** *noWhile.simps*(*7*) [*simp del*]

**lemma** *noWhileL-intro*[*intro*]:
**assumes** $\bigwedge$ *c*. $c \in$ *set cl* $\Longrightarrow$ *noWhile c*
**shows** *noWhileL cl*
⟨*proof*⟩

**lemma** *noWhileL-fin*[*simp*]:
**assumes** *noWhileL cl* **and** $c \in$ *set cl*
**shows** *noWhile c*
⟨*proof*⟩

**lemma** *noWhileL-update*[*simp*]:
**assumes** *cl*: *noWhileL cl* **and** *c'*: *noWhile c'*
**shows** *noWhileL* (*cl*[$n$ := *c'*])
⟨*proof*⟩

**definition** *finishedL* **where**
*finishedL cl* $\equiv \forall$ $c \in$ *set cl*. *finished c*

**lemma** *finished-Par-finishedL*[*simp*]:
*finished* (*Par cl*) $\longleftrightarrow$ *finishedL cl*
⟨*proof*⟩

**lemma** *finished-ParT-finishedL*[*simp*]:
*finished* (*ParT cl*) $\longleftrightarrow$ *finishedL cl*
⟨*proof*⟩

**declare** *finished.simps*(*6*) [*simp del*]
**declare** *finished.simps*(*7*) [*simp del*]

**lemma** *finishedL-intro*[*intro*]:
**assumes** $\bigwedge$ *c*. $c \in$ *set cl* $\Longrightarrow$ *finished c*
**shows** *finishedL cl*
⟨*proof*⟩

**lemma** *finishedL-finished*[*simp*]:
**assumes** *finishedL cl* **and** $c \in$ *set cl*
**shows** *finished c*
⟨*proof*⟩

**lemma** *finishedL-update*[*simp*]:
**assumes** *cl*: *finishedL cl* **and** $c'$: *finished* $c'$
**shows** *finishedL* ($cl[n := c']$)
⟨*proof*⟩

**lemma** *finished-fin*[*simp*]:
*finished c* $\Longrightarrow$ *noWhile c*
⟨*proof*⟩

**lemma** *finishedL-noWhileL*[*simp*]:
*finishedL cl* $\Longrightarrow$ *noWhileL cl*
⟨*proof*⟩

**locale** *PL* =
  **fixes**
    *aval* :: *'atom* ⇒ *'state* ⇒ *'state* **and**
    *tval* :: *'test* ⇒ *'state* ⇒ *bool* **and**
    *cval* :: *'choice* ⇒ *'state* ⇒ *real*
  **assumes**
    *properCh*: $\bigwedge$ *ch s*. $0 \leq$ *cval ch s* $\wedge$ *cval ch s* $\leq 1$
**begin**

**lemma** [*simp*]: (*n*::*nat*) < *N* $\Longrightarrow$ $0 \leq 1\ /\ N$ ⟨*proof*⟩

**lemma** [*simp*]: (*n*::*nat*) < *N* $\Longrightarrow$ $1\ /\ N \leq 1$ ⟨*proof*⟩

**lemma** [*simp*]: (*n*::*nat*) < *N* $\Longrightarrow$ $0 \leq 1 - 1\ /\ N$ ⟨*proof*⟩

**lemma** *sum-equal*: 0 < (*N*::*nat*) $\Longrightarrow$ *sum* ($\lambda$ *n*. *1*/*N*) {..< *N*} = *1*
⟨*proof*⟩

**fun** *proper* **where**

*proper Done* ⟷ *True*
| *proper* (*Atm x*) ⟷ *True*
| *proper* (*Seq c1 c2*) ⟷ *proper c1* ∧ *proper c2*
| *proper* (*While tst c*) ⟷ *proper c*
| *proper* (*Ch ch c1 c2*) ⟷ *proper c1* ∧ *proper c2*
| *proper* (*Par cl*) ⟷ *cl* ≠ [] ∧ (∀ *c* ∈ *set cl*. *proper c*)
| *proper* (*ParT cl*) ⟷ *cl* ≠ [] ∧ (∀ *c* ∈ *set cl*. *proper c*)

**definition** *properL* **where**
*properL cl* ≡ *cl* ≠ [] ∧ (∀ *c* ∈ *set cl*. *proper c*)

**lemma** *proper-Par-properL*[*simp*]:
*proper* (*Par cl*) ⟷ *properL cl*
⟨*proof*⟩

**lemma** *proper-ParT-properL*[*simp*]:
*proper* (*ParT cl*) ⟷ *properL cl*
⟨*proof*⟩

**declare** *proper.simps*(*6*) [*simp del*]
**declare** *proper.simps*(*7*) [*simp del*]

**lemma** *properL-intro*[*intro*]:
⟦*cl* ≠ []; ⋀ *c*. *c* ∈ *set cl* ⟹ *proper c*⟧ ⟹ *properL cl*
⟨*proof*⟩

**lemma** *properL-notEmp*[*simp*]: *properL cl* ⟹ *cl* ≠ []
⟨*proof*⟩

**lemma** *properL-proper*[*simp*]:
⟦*properL cl*; *c* ∈ *set cl*⟧ ⟹ *proper c*
⟨*proof*⟩

**lemma** *properL-update*[*simp*]:
**assumes** *cl*: *properL cl* **and** *c'*: *proper c'*
**shows** *properL* (*cl*[*n* := *c'*])
⟨*proof*⟩

**lemma** *proper-induct*[*consumes 1*, *case-names Done Atm Seq While Ch Par ParT*]:
**assumes** ∗: *proper c*
**and** *Done*: *phi Done*
**and** *Atm*: ⋀ *atm*. *phi* (*Atm atm*)
**and** *Seq*: ⋀ *c1 c2*. ⟦*phi c1*; *phi c2*⟧ ⟹ *phi* (*c1* ;; *c2*)
**and** *While*: ⋀ *tst c*. *phi c* ⟹ *phi* (*While tst c*)
**and** *Ch*: ⋀ *ch c1 c2*. ⟦*phi c1*; *phi c2*⟧ ⟹ *phi* (*Ch ch c1 c2*)
**and** *Par*: ⋀ *cl*. ⟦*properL cl*; ⋀ *c*. *c* ∈ *set cl* ⟹ *phi c*⟧ ⟹ *phi* (*Par cl*)
**and** *ParT*: ⋀ *cl*. ⟦*properL cl*; ⋀ *c*. *c* ∈ *set cl* ⟹ *phi c*⟧ ⟹ *phi* (*ParT cl*)
**shows** *phi c*
⟨*proof*⟩

### 1.2.1 Operational Small-Step Semantics

**definition** *theFT cl* ≡ $\{n.\ n < length\ cl \wedge finished\ (cl!n)\}$

**definition** *theNFT cl* ≡ $\{n.\ n < length\ cl \wedge \neg\ finished\ (cl!n)\}$

**lemma** *finite-theFT*[*simp*]: *finite* (*theFT cl*)
⟨*proof*⟩

**lemma** *theFT-length*[*simp*]: $n \in theFT\ cl \Longrightarrow n < length\ cl$
⟨*proof*⟩

**lemma** *theFT-finished*[*simp*]: $n \in theFT\ cl \Longrightarrow finished\ (cl!n)$
⟨*proof*⟩

**lemma** *finite-theNFT*[*simp*]: *finite* (*theNFT cl*)
⟨*proof*⟩

**lemma** *theNFT-length*[*simp*]: $n \in theNFT\ cl \Longrightarrow n < length\ cl$
⟨*proof*⟩

**lemma** *theNFT-notFinished*[*simp*]: $n \in theNFT\ cl \Longrightarrow \neg\ finished\ (cl!n)$
⟨*proof*⟩

**lemma** *theFT-Int-theNFT*[*simp*]:
*theFT cl Int theNFT cl* = {} **and** *theNFT cl Int theFT cl* = {}
⟨*proof*⟩

**lemma** *theFT-Un-theNFT*[*simp*]:
*theFT cl Un theNFT cl* = $\{..< length\ cl\}$ **and**
*theNFT cl Un theFT cl* = $\{..< length\ cl\}$
⟨*proof*⟩

**lemma** *in-theFT-theNFT*[*simp*]:
**assumes** $n1 \in theFT\ cl$ **and** $n2 \in theNFT\ cl$
**shows** $n1 \neq n2$ **and** $n2 \neq n1$
⟨*proof*⟩

**definition** *WtFT cl* ≡ *sum* (λ (*n*::*nat*). *1*/(*length cl*)) (*theFT cl*)

**definition** *WtNFT cl* ≡ *sum* (λ (*n*::*nat*). *1*/(*length cl*)) (*theNFT cl*)

**lemma** *WtFT-WtNFT*[*simp*]:
**assumes** $0 < length\ cl$
**shows** *WtFT cl* + *WtNFT cl* = *1* (**is** *?A* = *1*)
⟨*proof*⟩

**lemma** *WtNFT-1-WtFT*: $0 < length\ cl \Longrightarrow WtNFT\ cl = 1 - WtFT\ cl$
⟨*proof*⟩

**lemma** *WtNFT-WtFT-1*[*simp*]:
**assumes** $0 < length\ cl$ **and** $WtFT\ cl \neq 1$
**shows** $WtNFT\ cl\ /\ (1 - WtFT\ cl) = 1$ (**is** $?A\ /\ ?B = 1$)
⟨*proof*⟩

**lemma** *WtFT-ge-0*[*simp*]: $WtFT\ cl \geq 0$
⟨*proof*⟩

**lemma** *WtFT-le-1*[*simp*]: $WtFT\ cl \leq 1$ (**is** $?L \leq 1$)
⟨*proof*⟩

**lemma** *le-1-WtFT*[*simp*]: $0 \leq 1 - WtFT\ cl$ (**is** $0 \leq ?R$)
⟨*proof*⟩

**lemma** *WtFT-lt-1*[*simp*]: $WtFT\ cl \neq 1 \Longrightarrow WtFT\ cl < 1$
⟨*proof*⟩

**lemma** *lt-1-WtFT*[*simp*]: $WtFT\ cl \neq 1 \Longrightarrow 0 < 1 - WtFT\ cl$
⟨*proof*⟩

**lemma** *notFinished-WtFT*[*simp*]:
**assumes** $n < length\ cl$ **and** $\neg\ finished\ (cl\ !\ n)$
**shows** $1\ /\ length\ cl \leq 1 - WtFT\ cl$
⟨*proof*⟩

**fun** *brn* :: ('*test*, '*atom*, '*choice*) *cmd* ⇒ *nat* **where**
*brn Done* = *1*
| *brn* (*Atm atm*) = *1*
| *brn* (*c1* ;; *c2*) = *brn c1*
| *brn* (*While tst c*) = *1*
| *brn* (*Ch ch c1 c2*) = *2*
| *brn* (*Par cl*) = *lsum brn cl*
| *brn* (*ParT cl*) = *lsum brn cl*

**lemma** *brn-gt-0*: $proper\ c \Longrightarrow 0 < brn\ c$
⟨*proof*⟩

**lemma** *brn-gt-0-L*: $[\![properL\ cl;\ c \in set\ cl]\!] \Longrightarrow 0 < brn\ c$
⟨*proof*⟩

**definition** *locateT* ≡ *locate1 brn* **definition** *locateI* ≡ *locate2 brn*

**definition** *brnL cl n* ≡ *lsum brn* (*take n cl*)

**lemma** *brnL-lsum*: *brnL cl* (*length cl*) = *lsum brn cl*
⟨*proof*⟩

**lemma** *brnL-unique*:
**assumes** *properL cl* **and** *n1* < *length cl* ∧ *j1* < *brn* (*cl* ! *n1*)
**and** *n2* < *length cl* ∧ *j2* < *brn* (*cl* ! *n2*) **and** *brnL cl n1* + *j1* = *brnL cl n2* + *j2*
**shows** *n1* = *n2* ∧ *j1* = *j2*
⟨*proof*⟩

**lemma** *brn-Par-simp*[*simp*]: *brn* (*Par cl*) = *brnL cl* (*length cl*)
⟨*proof*⟩

**lemma** *brn-ParT-simp*[*simp*]: *brn* (*ParT cl*) = *brnL cl* (*length cl*)
⟨*proof*⟩

**declare** *brn.simps*(*6*)[*simp del*] **declare** *brn.simps*(*7*)[*simp del*]

**lemma** *brnL-0*[*simp*]: *brnL cl 0* = *0*
⟨*proof*⟩

**lemma** *brnL-Suc*[*simp*]: *n* < *length cl* ⟹ *brnL cl* (*Suc n*) = *brnL cl n* + *brn* (*cl* ! *n*)
⟨*proof*⟩

**lemma** *brnL-mono*[*simp*]: *n1* ≤ *n2* ⟹ *brnL cl n1* ≤ *brnL cl n2*
⟨*proof*⟩

**lemma** *brnL-mono2*[*simp*]:
**assumes** *p*: *properL cl* **and** *n*: *n1* < *n2* **and** *l*: *n2* ≤ *length cl*
**shows** *brnL cl n1* < *brnL cl n2* (**is** *?L* < *?R*)
⟨*proof*⟩

**lemma** *brn-index*[*simp*]:
**assumes** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**shows** *brnL cl n* + *i* < *brnL cl* (*length cl*) (**is** *?L* < *?R*)
⟨*proof*⟩

**lemma** *brnL-gt-0*[*simp*]: ⟦*properL cl*; *0* < *n*⟧ ⟹ *0* < *brnL cl n*
⟨*proof*⟩

**lemma** *locateTI*:
**assumes** *properL cl* **and** *ii* < *brn* (*Par cl*)
**shows**
*locateT cl ii* < *length cl* ∧
*locateI cl ii* < *brn* (*cl* ! (*locateT cl ii*)) ∧
*ii* = *brnL cl* (*locateT cl ii*) + *locateI cl ii*
⟨*proof*⟩

**lemma** *locateTI-unique*:

**assumes** *properL cl* **and** $n <$ *length cl*
**and** $i <$ *brn* (*cl* ! *n*) **and** *ii* = *brnL cl n* + *i*
**shows** *n* = *locateT cl ii* ∧ *i* = *locateI cl ii*
⟨*proof*⟩

**definition** *pickFT-pred* **where** *pickFT-pred cl n* ≡ $n <$ *length cl* ∧ *finished* (*cl*!*n*)
**definition** *pickFT* **where** *pickFT cl* ≡ *SOME n. pickFT-pred cl n*

**lemma** *pickFT-pred*:
**assumes** *WtFT cl* = *1* **shows** ∃ *n. pickFT-pred cl n*
⟨*proof*⟩

**lemma** *pickFT-pred-pickFT*: *WtFT cl* = *1* ⟹ *pickFT-pred cl* (*pickFT cl*)
⟨*proof*⟩

**lemma** *pickFT-length*[*simp*]: *WtFT cl* = *1* ⟹ *pickFT cl* < *length cl*
⟨*proof*⟩

**lemma** *pickFT-finished*[*simp*]: *WtFT cl* = *1* ⟹ *finished* (*cl* ! (*pickFT cl*))
⟨*proof*⟩

**lemma** *pickFT-theFT*[*simp*]: *WtFT cl* = *1* ⟹ *pickFT cl* ∈ *theFT cl*
⟨*proof*⟩

**fun** *wt-cont-eff* **where**
*wt-cont-eff Done s i* = (*1*, *Done*, *s*)
|
*wt-cont-eff* (*Atm atm*) *s i* = (*1*, *Done*, *aval atm s*)
|
*wt-cont-eff* (*c1* ;; *c2*) *s i* =
(*case wt-cont-eff c1 s i of*
(*x*, *c1′*, *s′*) ⇒
*if finished c1′ then* (*x*, *c2*, *s′*) *else* (*x*, *c1′* ;; *c2*, *s′*))
|
*wt-cont-eff* (*While tst c*) *s i* =
(*if tval tst s*
*then* (*1*, *c* ;; (*While tst c*), *s*)
*else* (*1*, *Done*, *s*))
|
*wt-cont-eff* (*Ch ch c1 c2*) *s i* =
(*if i* = *0 then cval ch s else 1* − *cval ch s*,
*if i* = *0 then c1 else c2*,
*s*)
|
*wt-cont-eff* (*Par cl*) *s ii* =
(*if cl* ! (*locateT cl ii*) ∈ *set cl then*
(*case wt-cont-eff*

```
          (cl ! (locateT cl ii))
          s
          (locateI cl ii) of
     (w, c', s') ⇒
       ((1 / (length cl)) ∗ w,
        Par (cl [(locateT cl ii) := c']),
        s'))
  else undefined)
|
wt-cont-eff (ParT cl) s ii =
 (if cl ! (locateT cl ii) ∈ set cl
   then
     (case wt-cont-eff
              (cl ! (locateT cl ii))
              s
              (locateI cl ii) of
         (w, c', s') ⇒
            (if WtFT cl = 1
             then (if locateT cl ii = pickFT cl ∧ locateI cl ii = 0
                      then 1
                      else 0)
             else if finished (cl ! (locateT cl ii))
               then 0
               else (1 / (length cl))
                    / (1 − WtFT cl)
                    ∗ w,
             ParT (cl [(locateT cl ii) := c']),
             s'))
   else undefined)
```

**definition** *wt* **where** *wt c s i = fst (wt-cont-eff c s i)*
**definition** *cont* **where** *cont c s i = fst (snd (wt-cont-eff c s i))*
**definition** *eff* **where** *eff c s i = snd (snd (wt-cont-eff c s i))*

**lemma** *wt-Done*[*simp*]: *wt Done s i = 1*
⟨*proof*⟩

**lemma** *wt-Atm*[*simp*]: *wt (Atm atm) s i = 1*
⟨*proof*⟩

**lemma** *wt-Seq*[*simp*]:
*wt (c1 ;; c2) s = wt c1 s*
⟨*proof*⟩

**lemma** *wt-While*[*simp*]: *wt (While tst c) s i = 1*
⟨*proof*⟩

**lemma** *wt-Ch-L*[*simp*]: *wt* (*Ch ch c1 c2*) *s 0* = *cval ch s*
⟨*proof*⟩

**lemma** *wt-Ch-R*[*simp*]: *wt* (*Ch ch c1 c2*) *s* (*Suc n*) = *1* − *cval ch s*
⟨*proof*⟩

**lemma** *cont-Done*[*simp*]: *cont Done s i* = *Done*
⟨*proof*⟩

**lemma** *cont-Atm*[*simp*]: *cont* (*Atm atm*) *s i* = *Done*
⟨*proof*⟩

**lemma** *cont-Seq-finished*[*simp*]: *finished* (*cont c1 s i*) ⟹ *cont* (*c1* ;; *c2*) *s i* = *c2*
⟨*proof*⟩

**lemma** *cont-Seq-notFinished*[*simp*]:
**assumes** ¬ *finished* (*cont c1 s i*)
**shows** *cont* (*c1* ;; *c2*) *s i* = (*cont c1 s i*) ;; *c2*
⟨*proof*⟩

**lemma** *cont-Seq-not-eq-finished*[*simp*]: ¬ *finished c2* ⟹ ¬ *finished* (*cont* (*Seq c1 c2*) *s i*)
⟨*proof*⟩

**lemma** *cont-While-False*[*simp*]: *tval tst s* = *False* ⟹ *cont* (*While tst c*) *s i* = *Done*
⟨*proof*⟩

**lemma** *cont-While-True*[*simp*]: *tval tst s* = *True* ⟹ *cont* (*While tst c*) *s i* = *c* ;; (*While tst c*)
⟨*proof*⟩

**lemma** *cont-Ch-L*[*simp*]: *cont* (*Ch ch c1 c2*) *s 0* = *c1*
⟨*proof*⟩

**lemma** *cont-Ch-R*[*simp*]: *cont* (*Ch ch c1 c2*) *s* (*Suc n*) = *c2*
⟨*proof*⟩

**lemma** *eff-Done*[*simp*]: *eff Done s i* = *s*
⟨*proof*⟩

**lemma** *eff-Atm*[*simp*]: *eff* (*Atm atm*) *s i* = *aval atm s*
⟨*proof*⟩

**lemma** *eff-Seq*[*simp*]: *eff* (*c1* ;; *c2*) *s* = *eff c1 s*
⟨*proof*⟩

**lemma** *eff-While*[*simp*]: *eff* (*While tst c*) *s i* = *s*
⟨*proof*⟩

**lemma** *eff-Ch*[*simp*]: *eff* (*Ch ch c1 c2*) *s i* = *s*
⟨*proof*⟩

**lemma** *brnL-nchotomy*:
**assumes** *properL cl* **and** *ii* < *brnL cl* (*length cl*)
**shows** ∃ *n i*. *n* < *length cl* ∧ *i* < *brn* (*cl* ! *n*) ∧ *ii* = *brnL cl n* + *i*
⟨*proof*⟩

**corollary** *brnL-cases*[*consumes 2*, *case-names Local*, *elim*]:
**assumes** *properL cl* **and** *ii* < *brnL cl* (*length cl*) **and**
⋀ *n i*. ⟦*n* < *length cl*; *i* < *brn* (*cl* ! *n*); *ii* = *brnL cl n* + *i*⟧ ⟹ *phi*
**shows** *phi*
⟨*proof*⟩

**lemma** *wt-cont-eff-Par*[*simp*]:
**assumes** *p*: *properL cl*
**and** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**shows**
*wt* (*Par cl*) *s* (*brnL cl n* + *i*) =
 *1* / (*length cl*) ∗ *wt* (*cl* ! *n*) *s i*
(**is** *?wL* = *?wR*)

*cont* (*Par cl*) *s* (*brnL cl n* + *i*) =
 *Par* (*cl* [*n* := *cont* (*cl* ! *n*) *s i*])
(**is** *?mL* = *?mR*)

*eff* (*Par cl*) *s* (*brnL cl n* + *i*) =
 *eff* (*cl* ! *n*) *s i*
(**is** *?eL* = *?eR*)
⟨*proof*⟩

**lemma** *cont-eff-ParT*[*simp*]:
**assumes** *p*: *properL cl*
**and** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**shows**
*cont* (*ParT cl*) *s* (*brnL cl n* + *i*) =
 *ParT* (*cl* [*n* := *cont* (*cl* ! *n*) *s i*])
(**is** *?mL* = *?mR*)

*eff* (*ParT cl*) *s* (*brnL cl n* + *i*) =
 *eff* (*cl* ! *n*) *s i*
(**is** *?eL* = *?eR*)
⟨*proof*⟩

**lemma** *wt-ParT-WtFT-pickFT-0*[*simp*]:
**assumes** *p*: *properL cl* **and** *WtFT*: *WtFT cl* = *1*
**shows** *wt* (*ParT cl*) *s* (*brnL cl* (*pickFT cl*)) = *1*
(**is** *?wL* = *1*)
⟨*proof*⟩

**lemma** *wt-ParT-WtFT-notPickFT-0*[*simp*]:
**assumes** *p*: *properL cl* **and** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**and** *WtFT*: *WtFT cl* = *1* **and** *ni*: *n* = *pickFT cl* ⟶ *i* ≠ *0*
**shows** *wt* (*ParT cl*) *s* (*brnL cl n* + *i*) = *0* (**is** *?wL* = *0*)
⟨*proof*⟩

**lemma** *wt-ParT-notWtFT-finished*[*simp*]:
**assumes** *p*: *properL cl* **and** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**and** *WtFT*: *WtFT cl* ≠ *1* **and** *f*: *finished* (*cl* ! *n*)
**shows** *wt* (*ParT cl*) *s* (*brnL cl n* + *i*) = *0* (**is** *?wL* = *0*)
⟨*proof*⟩

**lemma** *wt-cont-eff-ParT-notWtFT-notFinished*[*simp*]:
**assumes** *p*: *properL cl* **and** *n*: *n* < *length cl* **and** *i*: *i* < *brn* (*cl* ! *n*)
**and** *WtFT*: *WtFT cl* ≠ *1* **and** *nf*: ¬ *finished* (*cl* ! *n*)
**shows** *wt* (*ParT cl*) *s* (*brnL cl n* + *i*) =
    (*1* / (*length cl*)) / (*1* − *WtFT cl*) ∗ *wt* (*cl* ! *n*) *s i* (**is** *?wL* = *?wR*)
⟨*proof*⟩

**lemma** *wt-ge-0*[*simp*]:
**assumes** *proper c* **and** *i* < *brn c*
**shows** *0* ≤ *wt c s i*
⟨*proof*⟩

**lemma** *wt-le-1*[*simp*]:
**assumes** *proper c* **and** *i* < *brn c*
**shows** *wt c s i* ≤ *1*
⟨*proof*⟩

**abbreviation** *fromPlus* (‹(*1*{-..<+-})›) **where**
{*a* ..<+ *b*} ≡ {*a* ..< *a* + *b*}

**lemma** *brnL-UN*:
**assumes** *properL cl*
**shows** {..< *brnL cl* (*length cl*)} = (⋃ *n* < *length cl*. {*brnL cl n* ..<+ *brn* (*cl*!*n*)})
(**is** *?L* = (⋃ *n* < *length cl*. *?R n*))
⟨*proof*⟩

**lemma** *brnL-Int-lt*:
**assumes** *n12*: *n1* < *n2* **and** *n2*: *n2* < *length cl*
**shows**
{*brnL cl n1* ..<+ *brn* (*cl*!*n1*)} ∩ {*brnL cl n2* ..<+ *brn* (*cl*!*n2*)} = {}

⟨*proof*⟩

**lemma** *brnL-Int*:
**assumes** *n1* ≠ *n2* **and** *n1* < *length cl* **and** *n2* < *length cl*
**shows** {*brnL cl n1* ..<+ *brn* (*cl*!*n1*)} ∩ {*brnL cl n2* ..<+ *brn* (*cl*!*n2*)} = {}
⟨*proof*⟩

**lemma** *sum-wt-Par-sub*[*simp*]:
**assumes** *cl*: *properL cl* **and** *n*: *n* < *length cl* **and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows** *sum* (*wt* (*Par cl*) *s*) ((+) (*brnL cl n*) ' *I*) =
*1* /(*length cl*) ∗ *sum* (*wt* (*cl* ! *n*) *s*) *I* (**is** *?L* = *?wSch* ∗ *?R*)
⟨*proof*⟩

**lemma** *sum-wt-Par*[*simp*]:
**assumes** *cl*: *properL cl* **and** *n*: *n* < *length cl*
**shows** *sum* (*wt* (*Par cl*) *s*) {*brnL cl n* ..<+ *brn* (*cl*!*n*)} =
*1* /(*length cl*) ∗ *sum* (*wt* (*cl* ! *n*) *s*) {..< *brn* (*cl* ! *n*)} (**is** *?L* = *?W* ∗ *?R*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-WtFT-pickFT-0*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1*
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! (*pickFT cl*))} *0* ∈ *I*
**shows** *sum* (*wt* (*ParT cl*) *s*) ((+) (*brnL cl* (*pickFT cl*)) ' *I*) = *1* (**is** *?L* = *1*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-WtFT-pickFT-0-2*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1*
**and** *II*: *II* ⊆ {..< *brnL cl* (*length cl*)} *brnL cl* (*pickFT cl*) ∈ *II*
**shows** *sum* (*wt* (*ParT cl*) *s*) *II* = *1* (**is** *?L* = *1*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-WtFT-notPickFT-0*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1* **and** *n*: *n* < *length cl*
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)} **and** *nI*: *n* = *pickFT cl* ⟶ *0* ∉ *I*
**shows** *sum* (*wt* (*ParT cl*) *s*) ((+) (*brnL cl n*) ' *I*) = *0* (**is** *?L* = *0*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-notWtFT-finished*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* ≠ *1*
**and** *n*: *n* < *length cl* **and** *cln*: *finished* (*cl*!*n*) **and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows** *sum* (*wt* (*ParT cl*) *s*) ((+) (*brnL cl n*) ' *I*) = *0* (**is** *?L* = *0*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-notWtFT-notFinished*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* ≠ *1* **and** *n*: *n* < *length cl*
**and** *cln*: ¬ *finished* (*cl*!*n*) **and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows**
*sum* (*wt* (*ParT cl*) *s*) ((+) (*brnL cl n*) ' *I*) =
(*1* / (*length cl*)) / (*1* − *WtFT cl*) ∗ *sum* (*wt* (*cl* ! *n*) *s*) *I*

(**is** *?L* = *?w* / (*1* − *?wF*) ∗ *?R*)
⟨*proof*⟩

**lemma** *sum-wt-ParT-WtFT-pickFT-0*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1*
**shows** *sum* (*wt* (*ParT cl*) *s*) {*brnL cl* (*pickFT cl*) ..<+ *brn* (*cl* ! (*pickFT cl*))} = *1*
⟨*proof*⟩

**lemma** *sum-wt-ParT-WtFT-notPickFT-0*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1* **and** *n*: *n* < *length cl n* ≠ *pickFT cl*
**shows** *sum* (*wt* (*ParT cl*) *s*) {*brnL cl n* ..<+ *brn* (*cl*!*n*)} = *0*
⟨*proof*⟩

**lemma** *sum-wt-ParT-notWtFT-finished*[*simp*]:
**assumes** *cl*: *properL cl* **and** *WtFT cl* ≠ *1*
**and** *n*: *n* < *length cl* **and** *cln*: *finished* (*cl*!*n*)
**shows** *sum* (*wt* (*ParT cl*) *s*) {*brnL cl n* ..<+ *brn* (*cl*!*n*)} = *0*
⟨*proof*⟩

**lemma** *sum-wt-ParT-notWtFT-notFinished*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* ≠ *1*
**and** *n*: *n* < *length cl* **and** *cln*: ¬ *finished* (*cl*!*n*)
**shows**
*sum* (*wt* (*ParT cl*) *s*) {*brnL cl n* ..<+ *brn* (*cl*!*n*)} =
(*1* / (*length cl*)) / (*1* − *WtFT cl*) ∗
*sum* (*wt* (*cl* ! *n*) *s*) {..< *brn* (*cl* ! *n*)}
⟨*proof*⟩

**lemma** *sum-wt*[*simp*]:
**assumes** *proper c*
**shows** *sum* (*wt c s*) {..< *brn c*} = *1*
⟨*proof*⟩

**lemma** *proper-cont*[*simp*]:
**assumes** *proper c* **and** *i* < *brn c*
**shows** *proper* (*cont c s i*)
⟨*proof*⟩

**lemma** *sum-subset-le-1*[*simp*]:
**assumes** ∗: *proper c* **and** ∗∗: *I* ⊆ {..< *brn c*}
**shows** *sum* (*wt c s*) *I* ≤ *1*
⟨*proof*⟩

**lemma** *sum-le-1*[*simp*]:
**assumes** ∗: *proper c* **and** ∗∗: *i* < *brn c*
**shows** *sum* (*wt c s*) {..*i*} ≤ *1*
⟨*proof*⟩

### 1.2.2 Operations on configurations

**definition** *cont-eff cf b* = *snd* (*wt-cont-eff* (*fst cf*) (*snd cf*) *b*)

**lemma** *cont-eff*: *cont-eff cf b* = (*cont* (*fst cf*) (*snd cf*) *b*, *eff* (*fst cf*) (*snd cf*) *b*)
⟨*proof*⟩

**end**

**end**

# 2 Resumption-Based Noninterference

**theory** *Resumption-Based*
**imports** *Language-Semantics*
**begin**

**type-synonym** $'a$ *rel* = $('a \times 'a)$ *set*

## 2.1 Preliminaries

**lemma** *int-emp*[*simp*]:
**assumes** $i > 0$
**shows** $\{..<i\} \neq \{\}$
⟨*proof*⟩

**lemma** *inj-on-inv-into*[*simp*]:
**assumes** *inj-on F P*
**shows** *inv-into P F* ‘ (*F* ‘ *P*) = *P*
⟨*proof*⟩

**lemma** *inj-on-inv-into2*[*simp*]:
*inj-on* (*inv-into P F*) (*F* ‘ *P*)
⟨*proof*⟩

**lemma** *refl-gfp*:
**assumes** *1*: *mono Retr* **and** *2*: $\bigwedge$ *theta*. *refl theta* $\Longrightarrow$ *refl* (*Retr theta*)
**shows** *refl* (*gfp Retr*)
⟨*proof*⟩

**lemma** *sym-gfp*:
**assumes** *1*: *mono Retr* **and** *2*: $\bigwedge$ *theta*. *sym theta* $\Longrightarrow$ *sym* (*Retr theta*)
**shows** *sym* (*gfp Retr*)
⟨*proof*⟩

**lemma** *trancl-trans*[*simp*]:

**assumes** *trans R*
**shows** $P \hat{}+ \subseteq R \longleftrightarrow P \subseteq R$
⟨*proof*⟩

**lemma** *trans-gfp*:
**assumes** *1*: *mono Retr* **and** *2*: $\bigwedge$ *theta. trans theta* $\Longrightarrow$ *trans* (*Retr theta*)
**shows** *trans* (*gfp Retr*)
⟨*proof*⟩

**lemma** *O-subset-trans*:
**assumes** $r\ O\ r \subseteq r$
**shows** *trans r*
⟨*proof*⟩

**lemma** *trancl-imp-trans*:
**assumes** $r \hat{}+ \subseteq r$
**shows** *trans r*
⟨*proof*⟩

**lemma** *sym-trans-gfp*:
**assumes** *1*: *mono Retr* **and** *2*: $\bigwedge$ *theta. sym theta* $\wedge$ *trans theta* $\Longrightarrow$ *sym* (*Retr theta*) $\wedge$ *trans* (*Retr theta*)
**shows** *sym* (*gfp Retr*) $\wedge$ *trans* (*gfp Retr*)
⟨*proof*⟩

## 2.2 Infrastructure for partitions

**definition** *part* **where**
*part J P* $\equiv$
*Union P* $= J \wedge$
$(\forall\ J1\ J2.\ J1 \in P \wedge J2 \in P \wedge J1 \neq J2 \longrightarrow J1 \cap J2 = \{\})$

**inductive-set** *gen*
**for** $P$ :: *'a set set* **and** $I$ :: *'a set* **where**
*incl*[*simp*]: $i \in I \Longrightarrow i \in$ *gen P I*
|*ext*[*simp*]: $[\![J \in P;\ j0 \in J;\ j0 \in gen\ P\ I;\ j \in J]\!] \Longrightarrow j \in gen\ P\ I$

**definition** *partGen* **where**
*partGen P* $\equiv \{gen\ P\ I \mid I\ .\ I \in P\}$

**definition** *finer* **where**
*finer P Q* $\equiv$
$(\forall\ J \in Q.\ J = Union\ \{I \in P\ .\ I \subseteq J\}) \wedge$
$(P \neq \{\} \longrightarrow Q \neq \{\})$

**definition** *partJoin* **where**
*partJoin P Q* ≡ *partGen* (*P* ∪ *Q*)

**definition** *compat* **where**
*compat I theta f* ≡ ∀ *i j*. {*i*, *j*} ⊆ *I* ∧ *i* ≠ *j* ⟶ (*f i*, *f j*) ∈ *theta*

**definition** *partCompat* **where**
*partCompat P theta f* ≡
∀ *I* ∈ *P*. *compat I theta f*

**definition** *lift* **where**
*lift P F II* ≡ *Union* {*F I* | *I* . *I* ∈ *P* ∧ *I* ⊆ *II*}

part:

**lemma** *part-emp*[*simp*]:
*part J* (*insert* {} *P*) = *part J P*
⟨*proof*⟩

**lemma** *finite-part*[*simp*]:
**assumes** *finite I* **and** *part I P*
**shows** *finite P*
⟨*proof*⟩

**lemma** *part-sum*:
  **assumes** *P*: *part* {..<*n*::*nat*} *P*
  **shows** ($\sum$ *i*<*n*. *f i*) = ($\sum$ *p*∈*P*. $\sum$ *i*∈*p*. *f i*)
⟨*proof*⟩

**lemma** *part-Un*[*simp*]:
**assumes** *part I1 P1* **and** *part I2 P2* **and** *I1 Int I2* = {}
**shows** *part* (*I1 Un I2*) (*P1 Un P2*)
  ⟨*proof*⟩

**lemma** *part-Un-singl*[*simp*]:
**assumes** *part K P* **and** ⋀ *I*. *I* ∈ *P* ⟹ *I0 Int I* = {}
**shows** *part* (*I0 Un K*) ({*I0*} *Un P*)
⟨*proof*⟩

**lemma** *part-Un-singl2*:
**assumes** *K01* = *I0 Un K1*
**and** *part K1 P* **and** ⋀ *I*. *I* ∈ *P* ⟹ *I0 Int I* = {}
**shows** *part K01* ({*I0*} *Un P*)
⟨*proof*⟩

**lemma** *part-UN*:
**assumes** ⋀ *n*. *n* ∈ *N* ⟹ *part* (*I n*) (*P n*)

**and** $\bigwedge$ *n1 n2*. {*n1*,*n2*} ⊆ *N* ∧ *n1* ≠ *n2* ⟹ *I n1* ∩ *I n2* = {}
**shows** *part* (*UN n* : *N*. *I n*) (*UN n* : *N*. *P n*)
⟨*proof*⟩

gen:

**lemma** *incl-gen*[*simp*]:
*I* ⊆ *gen P I*
⟨*proof*⟩

**lemma** *gen-incl-Un*:
*gen P I* ⊆ *I* ∪ (*Union P*)
⟨*proof*⟩

**lemma** *gen-incl*:
**assumes** *I* ∈ *P*
**shows** *gen P I* ⊆ *Union P*
⟨*proof*⟩

**lemma** *finite-gen*:
**assumes** *finite P* **and** $\bigwedge$ *J*. *J* ∈ *P* ⟹ *finite J* **and** *finite I*
**shows** *finite* (*gen P I*)
⟨*proof*⟩

**lemma** *subset-gen*[*simp*]:
**assumes** *J* ∈ *P* **and** *gen P I* ∩ *J* ≠ {}
**shows** *J* ⊆ *gen P I*
⟨*proof*⟩

**lemma** *gen-subset-gen*[*simp*]:
**assumes** *J* ∈ *P* **and** *gen P I* ∩ *J* ≠ {}
**shows** *gen P J* ⊆ *gen P I*
⟨*proof*⟩

**lemma** *gen-mono*[*simp*]:
**assumes** *I* ⊆ *J*
**shows** *gen P I* ⊆ *gen P J*
⟨*proof*⟩

**lemma** *gen-idem*[*simp*]:
*gen P* (*gen P I*) = *gen P I*
⟨*proof*⟩

**lemma** *gen-nchotomy*:
**assumes** *J* ∈ *P*
**shows** *J* ⊆ *gen P I* ∨ *gen P I* ∩ *J* = {}
⟨*proof*⟩

**lemma** *gen-Union*:
**assumes** *I* ∈ *P*

**shows** *gen P I* = *Union* {*J* ∈ *P* . *J* ⊆ *gen P I*}
⟨*proof*⟩

**lemma** *subset-gen2*:
**assumes** ∗: {*I*,*J*} ⊆ *P* **and** ∗∗: *gen P I* ∩ *gen P J* ≠ {}
**shows** *I* ⊆ *gen P J*
⟨*proof*⟩

**lemma** *gen-subset-gen2*[*simp*]:
**assumes** {*I*,*J*} ⊆ *P* **and** *gen P I* ∩ *gen P J* ≠ {}
**shows** *gen P I* ⊆ *gen P J*
⟨*proof*⟩

**lemma** *gen-eq-gen*:
**assumes** {*I*,*J*} ⊆ *P* **and** *gen P I* ∩ *gen P J* ≠ {}
**shows** *gen P I* = *gen P J*
⟨*proof*⟩

**lemma** *gen-empty*[*simp*]:
*gen P* {} = {}
⟨*proof*⟩

**lemma** *gen-empty2*[*simp*]:
*gen* {} *I* = *I*
⟨*proof*⟩

**lemma** *emp-gen*[*simp*]:
**assumes** *gen P I* = {}
**shows** *I* = {}
⟨*proof*⟩

partGen:

**lemma** *partGen-ex*:
**assumes** *I* ∈ *P*
**shows** ∃ *J* ∈ *partGen P*. *I* ⊆ *J*
⟨*proof*⟩

**lemma** *ex-partGen*:
**assumes** *J* ∈ *partGen P* **and** *j*: *j* ∈ *J*
**shows** ∃ *I* ∈ *P*. *j* ∈ *I*
⟨*proof*⟩

**lemma** *Union-partGen*: ⋃ (*partGen P*) = ⋃ *P*
⟨*proof*⟩

**lemma** *Int-partGen*:
**assumes** ∗: {*I*,*J*} ⊆ *partGen P* **and** ∗∗: *I* ∩ *J* ≠ {}
**shows** *I* = *J*
⟨*proof*⟩

**lemma** *part-partGen*:
*part* (*Union P*) (*partGen P*)
⟨*proof*⟩

**lemma** *finite-partGen*[*simp*]:
**assumes** *finite P*
**shows** *finite* (*partGen P*)
⟨*proof*⟩

**lemma** *emp-partGen*[*simp*]:
**assumes** $\{\} \notin P$
**shows** $\{\} \notin$ *partGen P*
⟨*proof*⟩

finer:

**lemma** *finer-partGen*:
*finer P* (*partGen P*)
⟨*proof*⟩

**lemma** *finer-nchotomy*:
**assumes** *P*: *part I0 P* **and** *Q*: *part I0 Q* **and** *PQ*: *finer P Q*
**and** *I*: $I \in P$ **and** *II*: $II \in Q$
**shows** $I \subseteq II \vee (I \cap II = \{\})$
⟨*proof*⟩

**lemma** *finer-ex*:
**assumes** *P*: *part I0 P* **and** *Q*: *part I0 Q* **and** *PQ*: *finer P Q*
**and** *I*: $I \in P$
**shows** $\exists\ II.\ II \in Q \wedge I \subseteq II$
⟨*proof*⟩

partJoin:

**lemma** *partJoin-commute*:
*partJoin P Q* = *partJoin Q P*
⟨*proof*⟩

**lemma** *Union-partJoin-L*:
*Union P* ⊆ *Union* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *Union-partJoin-R*:
*Union Q* ⊆ *Union* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *part-partJoin*[*simp*]:
**assumes** *part I P* **and** *part I Q*
**shows** *part I* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *finer-partJoin-L*[*simp*]:
**assumes** ∗: *part I P* **and** ∗∗: *part I Q*
**shows** *finer P* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *finer-partJoin-R*[*simp*]:
**assumes** ∗: *part I P* **and** ∗∗: *part I Q*
**shows** *finer Q* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *finer-emp*[*simp*]:
**assumes** *finer* {} *Q*
**shows** $Q \subseteq$ { {} }
⟨*proof*⟩

compat:

**lemma** *part-emp-R*[*simp*]:
*part I* {} ⟷ *I* = {}
⟨*proof*⟩

**lemma** *part-emp-L*[*simp*]:
*part* {} *P* ⟹ *P* ⊆ { {} }
⟨*proof*⟩

**lemma** *finite-partJoin*[*simp*]:
**assumes** *finite P* **and** *finite Q*
**shows** *finite* (*partJoin P Q*)
⟨*proof*⟩

**lemma** *emp-partJoin*[*simp*]:
**assumes** {} ∉ *P* **and** {} ∉ *Q*
**shows** {} ∉ *partJoin P Q*
⟨*proof*⟩

partCompat:

**lemma** *partCompat-Un*[*simp*]:
*partCompat* (*P Un Q*) *theta f* ⟷
*partCompat P theta f* ∧ *partCompat Q theta f*
⟨*proof*⟩

**lemma** *partCompat-gen-aux*:
**assumes** *theta*: *sym theta trans theta*
**and** *fP*: *partCompat P theta f* **and** *I*: *I* ∈ *P*
**and** *i*: *i* ∈ *I* **and** *j*: *j* ∈ *gen P I* **and** *ij*: *i* ≠ *j*
**shows** (*f i*, *f j*) ∈ *theta*
⟨*proof*⟩

**lemma** *partCompat-gen*:

**assumes** *theta*: *sym theta trans theta*
**and** *fP*: *partCompat P theta f* **and** *I*: $I \in P$
**shows** *compat* (*gen P I*) *theta f*
⟨*proof*⟩

**lemma** *partCompat-partGen*:
**assumes** *sym theta* **and** *trans theta*
**and** *partCompat P theta f*
**shows** *partCompat* (*partGen P*) *theta f*
⟨*proof*⟩

**lemma** *partCompat-partJoin*[*simp*]:
**assumes** *sym theta* **and** *trans theta*
**and** *partCompat P theta f* **and** *partCompat Q theta f*
**shows** *partCompat* (*partJoin P Q*) *theta f*
⟨*proof*⟩

lift:

**lemma** *inj-on-lift*:
**assumes** *P*: *part I0 P* **and** *Q*: *part I0 Q* **and** *PQ*: *finer P Q*
**and** *F*: *inj-on F P* **and** *FP*: *part J0* (*F* ‘ *P*) **and** *emp*: $\{\} \notin F$ ‘ *P*
**shows** *inj-on* (*lift P F*) *Q*
⟨*proof*⟩

**lemma** *part-lift*:
**assumes** *P*: *part I0 P* **and** *Q*: *part I0 Q* **and** *PQ*: *finer P Q*
**and** *F*: *inj-on F P* **and** *FP*: *part J0* (*F* ‘ *P*) **and** *emp*: $\{\} \notin P$ $\{\} \notin F$ ‘ *P*
**shows** *part J0* (*lift P F* ‘ *Q*)
⟨*proof*⟩

**lemma** *finer-lift*:
**assumes** *finer P Q*
**shows** *finer* (*F* ‘ *P*) (*lift P F* ‘ *Q*)
⟨*proof*⟩

## 2.3 Basic setting for bisimilarity

**locale** *PL-Indis* =
  *PL aval tval cval*
  **for** *aval* :: *'atom* ⇒ *'state* ⇒ *'state* **and**
      *tval* :: *'test* ⇒ *'state* ⇒ *bool* **and**
      *cval* :: *'choice* ⇒ *'state* ⇒ *real* +
  **fixes**
    *indis* :: *'state rel*
  **assumes**
    *equiv-indis*: *equiv UNIV indis*

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

**no-notation** *eqpoll* (**infixl** ‹≈› *50*)

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

**lemma** *refl-indis*: *refl indis*
**and** *trans-indis*: *trans indis*
**and** *sym-indis*: *sym indis*
⟨*proof*⟩

**lemma** *indis-refl*[*intro*]: $s \approx s$
⟨*proof*⟩

**lemma** *indis-trans*[*trans*]: $[\![s \approx s'; s' \approx s'']\!] \Longrightarrow s \approx s''$
⟨*proof*⟩

**lemma** *indis-sym*[*sym*]: $s \approx s' \Longrightarrow s' \approx s$
⟨*proof*⟩

## 2.4 Discreetness

**coinductive** *discr* **where**
*intro*:
$(\bigwedge s\ i.\ i < brn\ c \longrightarrow s \approx eff\ c\ s\ i \wedge discr\ (cont\ c\ s\ i))$
$\Longrightarrow discr\ c$

**definition** *discrL* **where**
$discrL\ cl \equiv \forall\ c \in set\ cl.\ discr\ c$

**lemma** *discrL-intro*[*intro*]:
**assumes** $\bigwedge c.\ c \in set\ cl \Longrightarrow discr\ c$
**shows** *discrL cl*
⟨*proof*⟩

**lemma** *discrL-discr*[*simp*]:
**assumes** *discrL cl* **and** $c \in set\ cl$
**shows** *discr c*
⟨*proof*⟩

**lemma** *discrL-update*[*simp*]:
**assumes** *cl*: *discrL cl* **and** $c'$: $discr\ c'$
**shows** $discrL\ (cl[n := c'])$
⟨*proof*⟩

Coinduction for discreetness:

**lemma** *discr-coind*[*consumes 1*, *case-names Hyp*, *induct pred*: *discr*]:

**assumes** $*$: *phi c* **and**
$**$: $\bigwedge$ *c s i*. $[\![$ *phi c* ; $i <$ *brn c*$]\!]$
$\Longrightarrow s \approx$ *eff c s i* $\wedge$ (*phi* (*cont c s i*) $\vee$ *discr* (*cont c s i*))
**shows** *discr c*
⟨*proof*⟩

**lemma** *discr-raw-coind*[*consumes 1*, *case-names Hyp*]:
**assumes** $*$: *phi c* **and**
$**$: $\bigwedge$ *c s i*. $[\![i <$ *brn c*; *phi c*$]\!] \Longrightarrow s \approx$ *eff c s i* $\wedge$ *phi* (*cont c s i*)
**shows** *discr c*
⟨*proof*⟩

Discreetness versus transition:

**lemma** *discr-cont*[*simp*]:
**assumes** $*$: *discr c* **and** $**$: $i <$ *brn c*
**shows** *discr* (*cont c s i*)
⟨*proof*⟩

**lemma** *discr-eff-indis*[*simp*]:
**assumes** $*$: *discr c* **and** $**$: $i <$ *brn c*
**shows** $s \approx$ *eff c s i*
⟨*proof*⟩

## 2.5 Self-isomorphism

**coinductive** *siso* **where**
*intro*:
$[\![\bigwedge$ *s i*. $i <$ *brn c* $\Longrightarrow$ *siso* (*cont c s i*);
$\bigwedge$ *s t i*.
$i <$ *brn c* $\wedge\ s \approx t \Longrightarrow$
*eff c s i* $\approx$ *eff c t i* $\wedge$ *wt c s i* $=$ *wt c t i* $\wedge$ *cont c s i* $=$ *cont c t i*$]\!]$
$\Longrightarrow$ *siso c*

**definition** *sisoL* **where**
*sisoL cl* $\equiv \forall$ $c \in$ *set cl*. *siso c*

**lemma** *sisoL-intro*[*intro*]:
**assumes** $\bigwedge$ *c*. $c \in$ *set cl* $\Longrightarrow$ *siso c*
**shows** *sisoL cl*
⟨*proof*⟩

**lemma** *sisoL-siso*[*simp*]:
**assumes** *sisoL cl* **and** $c \in$ *set cl*
**shows** *siso c*
⟨*proof*⟩

**lemma** *sisoL-update*[*simp*]:
**assumes** *cl*: *sisoL cl* **and** $c'$: *siso* $c'$
**shows** *sisoL* (*cl*[$n := c'$])

⟨*proof*⟩

Coinduction for self-isomorphism:

**lemma** *siso-coind*[*consumes 1*, *case-names Obs Cont*, *induct pred*: *siso*]:
**assumes** ∗: *phi c* **and**
∗∗: $\bigwedge$ *c s t i*. ⟦*i* < *brn c*; *phi c*; *s* ≈ *t*⟧ ⟹
*eff c s i* ≈ *eff c t i* ∧ *wt c s i* = *wt c t i* ∧ *cont c s i* = *cont c t i* **and**
∗∗∗: $\bigwedge$ *c s i*. ⟦*i* < *brn c*; *phi c*⟧ ⟹ *phi* (*cont c s i*) ∨ *siso* (*cont c s i*)
**shows** *siso c*
⟨*proof*⟩

**lemma** *siso-raw-coind*[*consumes 1*, *case-names Obs Cont*]:
**assumes** ∗: *phi c* **and**
∗∗∗: $\bigwedge$ *c s t i*. ⟦*i* < *brn c*; *phi c*; *s* ≈ *t*⟧ ⟹
*eff c s i* ≈ *eff c t i* ∧ *wt c s i* = *wt c t i* ∧ *cont c s i* = *cont c t i* **and**
∗∗: $\bigwedge$ *c s i*. ⟦*i* < *brn c*; *phi c*⟧ ⟹ *phi* (*cont c s i*)
**shows** *siso c*
⟨*proof*⟩

Self-Isomorphism versus transition:

**lemma** *siso-cont*[*simp*]:
**assumes** ∗: *siso c* **and** ∗∗: *i* < *brn c*
**shows** *siso* (*cont c s i*)
⟨*proof*⟩

**lemma** *siso-cont-indis*[*simp*]:
**assumes** ∗: *siso c* **and** ∗∗: *s* ≈ *t i* < *brn c*
**shows** *eff c s i* ≈ *eff c t i* ∧ *wt c s i* = *wt c t i* ∧ *cont c s i* = *cont c t i*
⟨*proof*⟩

## 2.6 Notions of bisimilarity

Matchers

**definition** *mC-C-part* **where**
*mC-C-part c d P F* ≡
{} ∉ *P* ∧ {} ∉ *F* ‘ *P* ∧
*part* {..< *brn c*} *P* ∧ *part* {..< *brn d*} (*F* ‘ *P*)

**definition** *mC-C-wt* **where**
*mC-C-wt c d s t P F* ≡ ∀ *I* ∈ *P*. *sum* (*wt c s*) *I* = *sum* (*wt d t*) (*F I*)

**definition** *mC-C-eff-cont* **where**
*mC-C-eff-cont theta c d s t P F* ≡
∀ *I i j*.
*I* ∈ *P* ∧ *i* ∈ *I* ∧ *j* ∈ *F I* ⟶
*eff c s i* ≈ *eff d t j* ∧ (*cont c s i*, *cont d t j*) ∈ *theta*

**definition** *mC-C* **where**

*mC-C theta c d s t P F* ≡
*mC-C-part c d P F* ∧ *inj-on F P* ∧ *mC-C-wt c d s t P F* ∧ *mC-C-eff-cont theta c d s t P F*

**definition** *matchC-C* **where**
*matchC-C theta c d* ≡ ∀ *s t*. *s* ≈ *t* ⟶ (∃ *P F*. *mC-C theta c d s t P F*)

**definition** *mC-ZOC-part* **where**
*mC-ZOC-part c d s t I0 P F* ≡
{} ∉ *P* − {*I0*} ∧ {} ∉ *F* ‘ (*P* − {*I0*}) ∧ *I0* ∈ *P* ∧
*part* {..< *brn c*} *P* ∧ *part* {..< *brn d*} (*F* ‘ *P*)

**definition** *mC-ZOC-wt* **where**
*mC-ZOC-wt c d s t I0 P F* ≡
*sum* (*wt c s*) *I0* < *1* ∧ *sum* (*wt d t*) (*F I0*) < *1* ⟶
(∀ *I* ∈ *P* − {*I0*}.
*sum* (*wt c s*) *I* / (*1* − *sum* (*wt c s*) *I0*) =
*sum* (*wt d t*) (*F I*) / (*1* − *sum* (*wt d t*) (*F I0*)))

**definition** *mC-ZOC-eff-cont0* **where**
*mC-ZOC-eff-cont0 theta c d s t I0 F* ≡
(∀ *i* ∈ *I0*. *s* ≈ *eff c s i* ∧ (*cont c s i*, *d*) ∈ *theta*) ∧
(∀ *j* ∈ *F I0*. *t* ≈ *eff d t j* ∧ (*c*, *cont d t j*) ∈ *theta*)

**definition** *mC-ZOC-eff-cont* **where**
*mC-ZOC-eff-cont theta c d s t I0 P F* ≡
∀ *I i j*.
*I* ∈ *P* − {*I0*} ∧ *i* ∈ *I* ∧ *j* ∈ *F I* ⟶
*eff c s i* ≈ *eff d t j* ∧
(*cont c s i*, *cont d t j*) ∈ *theta*

**definition** *mC-ZOC* **where**
*mC-ZOC theta c d s t I0 P F* ≡
*mC-ZOC-part c d s t I0 P F* ∧
*inj-on F P* ∧
*mC-ZOC-wt c d s t I0 P F* ∧
*mC-ZOC-eff-cont0 theta c d s t I0 F* ∧
*mC-ZOC-eff-cont theta c d s t I0 P F*

**definition** *matchC-LC* **where**
*matchC-LC theta c d* ≡
∀ *s t*. *s* ≈ *t* ⟶ (∃ *I0 P F*. *mC-ZOC theta c d s t I0 P F*)

**lemmas** *m-defs* = *mC-C-def mC-ZOC-def*

**lemmas** *m-defsAll* =
*mC-C-def mC-C-part-def mC-C-wt-def mC-C-eff-cont-def*

*mC-ZOC-def mC-ZOC-part-def mC-ZOC-wt-def mC-ZOC-eff-cont0-def mC-ZOC-eff-cont-def*

**lemmas** *match-defs* =
*matchC-C-def matchC-LC-def*

**lemma** *mC-C-mono*:
**assumes** *mC-C theta c d s t P F* **and** *theta* ⊆ *theta′*
**shows** *mC-C theta′ c d s t P F*
⟨*proof*⟩

**lemma** *matchC-C-mono*:
**assumes** *matchC-C theta c d* **and** *theta* ⊆ *theta′*
**shows** *matchC-C theta′ c d*
⟨*proof*⟩

**lemma** *mC-ZOC-mono*:
**assumes** *mC-ZOC theta c d s t I0 P F* **and** *theta* ⊆ *theta′*
**shows** *mC-ZOC theta′ c d s t I0 P F*
⟨*proof*⟩

**lemma** *matchC-LC-mono*:
**assumes** *matchC-LC theta c d* **and** *theta* ⊆ *theta′*
**shows** *matchC-LC theta′ c d*
⟨*proof*⟩

**lemma** *Int-not-in-eq-emp*:
*P* ∩ {*I*. *I* ∉ *P*} = {}
⟨*proof*⟩

**lemma** *mC-C-mC-ZOC*:
**assumes** *mC-C theta c d s t P F*
**shows** *mC-ZOC theta c d s t* {} (*P Un* { {} }) (%*I*. *if I* ∈ *P then F I else* {})
(**is** *mC-ZOC theta c d s t ?I0 ?Q ?G*)
⟨*proof*⟩

**lemma** *matchC-C-matchC-LC*:
**assumes** *matchC-C theta c d*
**shows** *matchC-LC theta c d*
⟨*proof*⟩

Retracts:

**definition** *Sretr* **where**
*Sretr theta* ≡
{(*c*, *d*). *matchC-C theta c d*}

**definition** *ZOretr* **where**
*ZOretr theta* ≡
{(*c*,*d*). *matchC-LC theta c d*}

**lemmas** *Retr-defs* =
*Sretr-def*
*ZOretr-def*

**lemma** *mono-Retr*:
*mono Sretr*
*mono ZOretr*
⟨*proof*⟩

**lemma** *Retr-incl*:
*Sretr theta* ⊆ *ZOretr theta*
⟨*proof*⟩

The associated bisimilarity relations:

**definition** *Sbis* **where** *Sbis* ≡ *gfp Sretr*
**definition** *ZObis* **where** *ZObis* ≡ *gfp ZOretr*

**abbreviation** *Sbis-abbrev* (**infix** ‹≈*s*› *55*) **where** *c* ≈*s d* ≡ (*c*, *d*) : *Sbis*
**abbreviation** *ZObis-abbrev* (**infix** ‹≈*01*› *55*) **where** *c* ≈*01 d* ≡ (*c*, *d*) : *ZObis*

**lemmas** *bis-defs* = *Sbis-def ZObis-def*

**lemma** *bis-incl*:
*Sbis* ≤ *ZObis*
⟨*proof*⟩

**lemma** *bis-imp*[*simp*]:
⋀ *c1 c2*. *c1* ≈*s c2* ⟹ *c1* ≈*01 c2*
⟨*proof*⟩

**lemma** *Sbis-prefix*:
*Sbis* ≤ *Sretr Sbis*
⟨*proof*⟩

**lemma** *Sbis-fix*:
*Sretr Sbis* = *Sbis*
⟨*proof*⟩

**lemma** *Sbis-mC-C*:
**assumes** *c* ≈*s d* **and** *s* ≈ *t*
**shows** ∃ *P F*. *mC-C Sbis c d s t P F*
⟨*proof*⟩

**lemma** *Sbis-coind*:
**assumes** *theta* ≤ *Sretr* (*theta Un Sbis*)

**shows** *theta* $\leq$ *Sbis*
⟨*proof*⟩

**lemma** *Sbis-raw-coind*:
**assumes** *theta* $\leq$ *Sretr theta*
**shows** *theta* $\leq$ *Sbis*
⟨*proof*⟩

**lemma** *mC-C-sym*:
**assumes** *mC-C theta c d s t P F*
**shows** *mC-C* (*theta* ^−*1*) *d c t s* (*F* ' *P*) (*inv-into P F*)
⟨*proof*⟩

**lemma** *matchC-C-sym*:
**assumes** *matchC-C theta c d*
**shows** *matchC-C* (*theta* ^−*1*) *d c*
⟨*proof*⟩

**lemma** *Sretr-sym*:
**assumes** *sym theta*
**shows** *sym* (*Sretr theta*)
⟨*proof*⟩

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

**lemma** *Sbis-sym*: *c* $\approx$*s d* $\Longrightarrow$ *d* $\approx$*s c*
⟨*proof*⟩

**lemma** *mC-C-trans*:
**assumes** ∗: *mC-C theta1 c d s t P F* **and** ∗∗: *mC-C theta2 d e t u* (*F* ' *P*) *G*
**shows** *mC-C* (*theta1 O theta2*) *c e s u P* (*G o F*)
⟨*proof*⟩

**lemma** *mC-C-finer*:
**assumes** ∗: *mC-C theta c d s t P F*
**and** *theta*: *trans theta*
**and** *Q*: *finer P Q finite Q* {} $\notin$ *Q part* {..<*brn c*} *Q*
**and** *c*: *partCompat Q indis* (*eff c s*) *partCompat Q theta* (*cont c s*)
**shows** *mC-C theta c d s t Q* (*lift P F*)
⟨*proof*⟩

**lemma** *mC-C-partCompat-eff*:
**assumes** ∗: *mC-C theta c d s t P F*

**shows** *partCompat P indis* (*eff c s*)
⟨*proof*⟩

**lemma** *mC-C-partCompat-cont*:
**assumes** ∗: *mC-C theta c d s t P F*
**and** *theta*: *sym theta trans theta*
**shows** *partCompat P theta* (*cont c s*)
⟨*proof*⟩

**lemma** *matchC-C-sym-trans*:
**assumes** ∗: *matchC-C theta c1 c* **and** ∗∗: *matchC-C theta c c2*
**and** *theta*: *sym theta trans theta*
**shows** *matchC-C theta c1 c2*
⟨*proof*⟩

**lemma** *Sretr-sym-trans*:
**assumes** *sym theta* ∧ *trans theta*
**shows** *trans* (*Sretr theta*)
⟨*proof*⟩

**lemma** *trans-Sbis*: *trans Sbis*
⟨*proof*⟩

**lemma** *Sbis-trans*: ⟦*c* ≈*s d*; *d* ≈*s e*⟧ ⟹ *c* ≈*s e*
⟨*proof*⟩

**lemma** *ZObis-prefix*:
*ZObis* ≤ *ZOretr ZObis*
⟨*proof*⟩

**lemma** *ZObis-fix*:
*ZOretr ZObis* = *ZObis*
⟨*proof*⟩

**lemma** *ZObis-mC-ZOC*:
**assumes** *c* ≈*01 d* **and** *s* ≈ *t*
**shows** ∃ *I0 P F*. *mC-ZOC ZObis c d s t I0 P F*
⟨*proof*⟩

**lemma** *ZObis-coind*:
**assumes** *theta* ≤ *ZOretr* (*theta Un ZObis*)
**shows** *theta* ≤ *ZObis*
⟨*proof*⟩

**lemma** *ZObis-raw-coind*:
**assumes** *theta* ≤ *ZOretr theta*

**shows** *theta* $\leq$ *ZObis*
⟨*proof*⟩

**lemma** *mC-ZOC-sym*:
**assumes** *theta*: *sym theta* **and** $*$: *mC-ZOC theta c d s t I0 P F*
**shows** *mC-ZOC theta d c t s* (*F I0*) (*F ‘ P*) (*inv-into P F*)
⟨*proof*⟩

**lemma** *matchC-LC-sym*:
**assumes** $*$: *sym theta* **and** *matchC-LC theta c d*
**shows** *matchC-LC theta d c*
⟨*proof*⟩

**lemma** *ZOretr-sym*:
**assumes** *sym theta*
**shows** *sym* (*ZOretr theta*)
⟨*proof*⟩

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

**lemma** *ZObis-sym*: *c* $\approx$*01 d* $\Longrightarrow$ *d* $\approx$*01 c*
⟨*proof*⟩

## 2.7 List versions of the bisimilarities

**definition** *SbisL* **where**
*SbisL cl dl* $\equiv$
*length cl* = *length dl* $\wedge$ ($\forall$ *n* < *length cl*. *cl* ! *n* $\approx$*s dl* ! *n*)

**lemma** *SbisL-intro*[*intro*]:
**assumes** *length cl* = *length dl* **and**
$\bigwedge$ *n*. ⟦*n* < *length cl*; *n* < *length dl*⟧ $\Longrightarrow$ *cl* ! *n* $\approx$*s dl* ! *n*
**shows** *SbisL cl dl*
⟨*proof*⟩

**lemma** *SbisL-length*[*simp*]:
**assumes** *SbisL cl dl*
**shows** *length cl* = *length dl*
⟨*proof*⟩

**lemma** *SbisL-Sbis*[*simp*]:
**assumes** *SbisL cl dl* **and** *n* < *length cl* $\vee$ *n* < *length dl*
**shows** *cl* ! *n* $\approx$*s dl* ! *n*
⟨*proof*⟩

**lemma** *SbisL-update*[*simp*]:

**assumes** *cldl*: *SbisL cl dl* **and** *c'd'*: $c' \approx s\ d'$
**shows** *SbisL* (*cl* [$n := c'$]) (*dl* [$n := d'$]) (**is** *SbisL ?cl' ?dl'*)
⟨*proof*⟩

**lemma** *SbisL-update-L*[*simp*]:
**assumes** *SbisL cl dl* **and** $c' \approx s\ dl!n$
**shows** *SbisL* (*cl*[$n := c'$]) *dl*
⟨*proof*⟩

**lemma** *SbisL-update-R*[*simp*]:
**assumes** *SbisL cl dl* **and** $cl!n \approx s\ d'$
**shows** *SbisL cl* (*dl*[$n := d'$])
⟨*proof*⟩

**definition** *ZObisL* **where**
*ZObisL cl dl* ≡
*length cl* = *length dl* ∧ (∀ $n <$ *length cl*. *cl* ! $n \approx 01$ *dl* ! $n$)

**lemma** *ZObisL-intro*[*intro*]:
**assumes** *length cl* = *length dl* **and**
⋀ *n*. ⟦$n <$ *length cl*; $n <$ *length dl*⟧ ⟹ *cl* ! $n \approx 01$ *dl* ! $n$
**shows** *ZObisL cl dl*
⟨*proof*⟩

**lemma** *ZObisL-length*[*simp*]:
**assumes** *ZObisL cl dl*
**shows** *length cl* = *length dl*
⟨*proof*⟩

**lemma** *ZObisL-ZObis*[*simp*]:
**assumes** *ZObisL cl dl* **and** $n <$ *length cl* ∨ $n <$ *length dl*
**shows** *cl* ! $n \approx 01$ *dl* ! $n$
⟨*proof*⟩

**lemma** *ZObisL-update*[*simp*]:
**assumes** *cldl*: *ZObisL cl dl* **and** *c'd'*: $c' \approx 01\ d'$
**shows** *ZObisL* (*cl* [$n := c'$]) (*dl* [$n := d'$]) (**is** *ZObisL ?cl' ?dl'*)
⟨*proof*⟩

**lemma** *ZObisL-update-L*[*simp*]:
**assumes** *ZObisL cl dl* **and** $c' \approx 01\ dl!n$
**shows** *ZObisL* (*cl*[$n := c'$]) *dl*
⟨*proof*⟩

**lemma** *ZObisL-update-R*[*simp*]:
**assumes** *ZObisL cl dl* **and** $cl!n \approx 01\ d'$
**shows** *ZObisL cl* (*dl*[$n := d'$])

⟨*proof*⟩

## 2.8 Discreetness for configurations

**coinductive** *discrCf* **where**
*intro*:
$(\bigwedge i.\ i < brn\ (fst\ cf) \longrightarrow$
$\quad snd\ cf \approx snd\ (cont\text{-}eff\ cf\ i) \wedge discrCf\ (cont\text{-}eff\ cf\ i))$
$\Longrightarrow discrCf\ cf$

Coinduction for discrness:

**lemma** *discrCf-coind*[*consumes 1*, *case-names Hyp*, *induct pred*: *discr*]:
**assumes** ∗: *phi cf* **and**
∗∗: $\bigwedge cf\ i.$
$\quad [\![i < brn\ (fst\ cf);\ phi\ cf]\!] \Longrightarrow$
$\quad snd\ cf \approx snd\ (cont\text{-}eff\ cf\ i) \wedge (phi\ (cont\text{-}eff\ cf\ i) \vee discrCf\ (cont\text{-}eff\ cf\ i))$
**shows** *discrCf cf*
⟨*proof*⟩

**lemma** *discrCf-raw-coind*[*consumes 1*, *case-names Hyp*]:
**assumes** ∗: *phi cf* **and**
∗∗: $\bigwedge cf\ i.\ [\![i < brn\ (fst\ cf);\ phi\ cf]\!] \Longrightarrow$
$\quad snd\ cf \approx snd\ (cont\text{-}eff\ cf\ i) \wedge phi\ (cont\text{-}eff\ cf\ i)$
**shows** *discrCf cf*
⟨*proof*⟩

Discreetness versus transition:

**lemma** *discrCf-cont*[*simp*]:
**assumes** ∗: *discrCf cf* **and** ∗∗: $i < brn\ (fst\ cf)$
**shows** *discrCf* (*cont-eff cf i*)
⟨*proof*⟩

**lemma** *discrCf-eff-indis*[*simp*]:
**assumes** ∗: *discrCf cf* **and** ∗∗: $i < brn\ (fst\ cf)$
**shows** $snd\ cf \approx snd\ (cont\text{-}eff\ cf\ i)$
⟨*proof*⟩

**lemma** *discr-discrCf*:
**assumes** *discr c*
**shows** *discrCf* (*c*, *s*)
⟨*proof*⟩

**lemma** *ZObis-pres-discrCfL*:
**assumes** $fst\ cf \approx 01\ fst\ df$ **and** $snd\ cf \approx snd\ df$ **and** *discrCf df*
**shows** *discrCf cf*
⟨*proof*⟩

**corollary** *ZObis-pres-discrCfR*:
**assumes** *discrCf cf* **and** $fst\ cf \approx 01\ fst\ df$ **and** $snd\ cf \approx snd\ df$

**shows** *discrCf df*
⟨*proof*⟩

**end**

**end**

# 3 Trace-Based Noninterference

**theory** *Trace-Based*
**imports** *Resumption-Based*
**begin**

## 3.1 Preliminaries

**lemma** *dist-sum*:
**fixes** $f :: 'a \Rightarrow real$ **and** $g :: 'a \Rightarrow real$
**assumes** $\bigwedge i.\ i \in I \Longrightarrow dist\ (f\ i)\ (g\ i) \le e\ i$
**shows** $dist\ (\sum i{\in}I.\ f\ i)\ (\sum i{\in}I.\ g\ i) \le (\sum i{\in}I.\ e\ i)$
⟨*proof*⟩

**lemma** *dist-mult*[*simp*]: $dist\ (x * y)\ (x * z) = |x| * dist\ y\ z$
⟨*proof*⟩

**lemma** *dist-divide*[*simp*]: $dist\ (y\ /\ r)\ (z\ /\ r) = dist\ y\ z\ /\ |r|$
⟨*proof*⟩

**lemma** *dist-weighted-sum*:
**fixes** $f :: 'a \Rightarrow real$ **and** $g :: 'b \Rightarrow real$
**assumes** *eps*: $\bigwedge i\ j.\ i \in I \Longrightarrow j \in J \Longrightarrow w\ i \neq 0 \Longrightarrow v\ j \neq 0 \Longrightarrow dist\ (f\ i)\ (g\ j) \le d\ i + e\ j$
**and** *pos*: $\bigwedge i.\ i{\in}I \Longrightarrow 0 \le w\ i$ $\bigwedge j.\ j{\in}J \Longrightarrow 0 \le v\ j$
**and** *sum*: $(\sum i{\in}I.\ w\ i) = 1$ $(\sum j{\in}J.\ v\ j) = 1$
**shows** $dist\ (\sum i{\in}I.\ w\ i * f\ i)\ (\sum j{\in}J.\ v\ j * g\ j) \le (\sum i{\in}I.\ w\ i * d\ i) + (\sum j{\in}J.\ v\ j * e\ j)$
⟨*proof*⟩

**lemma** *field-abs-le-zero-epsilon*:
**fixes** $x :: 'a{::}\{linordered\text{-}field\}$
**assumes** *epsilon*: $\bigwedge e.\ 0 < e \Longrightarrow |x| \le e$
**shows** $|x| = 0$
⟨*proof*⟩

**lemma** *nat-nat-induct*[*case-names less*]:
**fixes** $P :: nat \Rightarrow nat \Rightarrow bool$
**assumes** *less*: $\bigwedge n\ m.\ (\bigwedge j\ k.\ j + k < n + m \Longrightarrow P\ j\ k) \Longrightarrow P\ n\ m$

**shows** *P n m*
⟨*proof*⟩

**lemma** *part-insert*:
**assumes** *part A P* **assumes** $X \cap A = \{\}$
**shows** *part* $(A \cup X)$ (*insert X P*)
⟨*proof*⟩

**lemma** *part-insert-subset*:
**assumes** *X*: *part* $(A - X)$ $P$ $X \subseteq A$
**shows** *part A* (*insert X P*)
⟨*proof*⟩

**lemma** *part-is-subset*:
*part S P* $\Longrightarrow p \in P \Longrightarrow p \subseteq S$
⟨*proof*⟩

**lemma** *dist-nonneg-bounded*:
**fixes** *l u x y* :: *real*
**assumes** $l \leq x$ $x \leq u$ $l \leq y$ $y \leq u$
**shows** *dist x y* $\leq u - l$
⟨*proof*⟩

**lemma** *integrable-count-space-finite-support*:
**fixes** *f* :: *'a* ⇒ *'b*::{*banach*,*second-countable-topology*}
**shows** *finite* $\{x{\in}X.\ f\ x \neq 0\} \Longrightarrow$ *integrable* (*count-space X*) *f*
⟨*proof*⟩

**lemma** *lebesgue-integral-point-measure*:
**fixes** *g* :: - ⇒ *real*
**assumes** *f*: *finite* $\{a{\in}A.\ 0 < f\ a \wedge g\ a \neq 0\}$
**shows** $integral^L$ (*point-measure A f*) $g = (\sum a|a{\in}A \wedge 0 < f\ a \wedge g\ a \neq 0.\ f\ a * g\ a)$

⟨*proof*⟩

**lemma** (**in** *finite-measure*) *finite-measure-dist*:
**assumes** *AE*: *AE x in M*. $x \notin C \longrightarrow (x \in A \longleftrightarrow x \in B)$
**assumes** *sets*: $A \in$ *sets M* $B \in$ *sets M* $C \in$ *sets M*
**shows** *dist* (*measure M A*) (*measure M B*) $\leq$ *measure M C*
⟨*proof*⟩

**lemma** (**in** *prob-space*) *prob-dist*:
**assumes** *AE*: *AE x in M*. $\neg$ *C x* $\longrightarrow$ (*A x* $\longleftrightarrow$ *B x*)
**assumes** *sets*: *Measurable.pred M A Measurable.pred M B Measurable.pred M C*
**shows** *dist* $\mathcal{P}$(*x in M*. *A x*) $\mathcal{P}$(*x in M*. *B x*) $\leq \mathcal{P}$(*x in M*. *C x*)
⟨*proof*⟩

**lemma** *Least-eq-0-iff*: $(\exists\, i{::}nat.\ P\ i) \Longrightarrow (LEAST\ i.\ P\ i) = 0 \longleftrightarrow P\ 0$

⟨*proof*⟩

**lemma** *case-nat-comp-Suc*[*simp*]: *case-nat x f* ∘ *Suc* = *f*
⟨*proof*⟩

**lemma** *sum-eq-0-iff*:
**fixes** *f* :: - ⇒ ′*a* :: {*comm-monoid-add*,*ordered-ab-group-add*}
**shows** *finite A* ⟹ (⋀*i*. *i* ∈ *A* ⟹ *0* ≤ *f i*) ⟹ (∑ *i*∈*A*. *f i*) = *0* ⟷ (∀ *i*∈*A*. *f i* = *0*)
⟨*proof*⟩

**lemma** *sum-less-0-iff*:
**fixes** *f* :: - ⇒ ′*a* :: {*comm-monoid-add*,*ordered-ab-group-add*}
**shows** *finite A* ⟹ (⋀*i*. *i* ∈ *A* ⟹ *0* ≤ *f i*) ⟹ *0* < (∑ *i*∈*A*. *f i*) ⟷ (∃ *i*∈*A*. *0* < *f i*)
⟨*proof*⟩

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

**declare** *emp-gen*[*simp del*]

**interpretation** *pmf-as-function* ⟨*proof*⟩

**lift-definition** *wt-pmf* :: (′*test*, ′*atom*, ′*choice*) *cmd* × ′*state* ⇒ *nat pmf* **is**
λ(*c*, *s*) *i*. *if proper c then if i* < *brn c then wt c s i else 0 else if i* = *0 then 1 else 0*
⟨*proof*⟩

**definition** *trans* :: (′*test*, ′*atom*, ′*choice*) *cmd* × ′*state* ⇒ ((′*test*, ′*atom*, ′*choice*) *cmd* × ′*state*) *pmf* **where**
*trans cf* = *map-pmf* (λ*i*. *if proper* (*fst cf*) *then cont-eff cf i else cf*) (*wt-pmf cf*)

**sublocale** *T?*: *MC-syntax trans* ⟨*proof*⟩

**abbreviation** *G cf* ≡ *set-pmf* (*trans cf*)

**lemma** *set-pmf-map*: *set-pmf* (*map-pmf f M*) = *f* ‘ *set-pmf M*
⟨*proof*⟩

**lemma** *set-pmf-wt*:
*set-pmf* (*wt-pmf cf*) = (*if proper* (*fst cf*) *then* {*i*. *i* < *brn* (*fst cf*) ∧ *0* < *wt* (*fst cf*) (*snd cf*) *i*} *else* {*0*})
⟨*proof*⟩

**lemma** *G-eq*:
*G cf* = (*if proper* (*fst cf*) *then* {*cont-eff cf i* | *i*. *i* < *brn* (*fst cf*) ∧ *0* < *wt* (*fst cf*) (*snd cf*) *i* } *else* {*cf*})
⟨*proof*⟩

**lemma** *discrCf-G*: *discrCf cf* $\Longrightarrow$ *cf*$'$ $\in$ *G cf* $\Longrightarrow$ *discrCf cf*$'$
⟨*proof*⟩

**lemma** *measurable-discrCf*[*measurable*]: *Measurable.pred* (*count-space UNIV*) *discrCf*
⟨*proof*⟩

**lemma** *measurable-indis*[*measurable*]: *Measurable.pred* (*count-space UNIV*) ($\lambda$*x*. *snd x* $\approx$ *c*)
⟨*proof*⟩

**lemma** *integral-trans*:
*proper* (*fst cf*) $\Longrightarrow$
($\int$ *x*. *f x* $\partial$*trans cf*) = ($\sum$ *i*<*brn* (*fst cf*). *wt* (*fst cf*) (*snd cf*) *i* $*$ *f* (*cont-eff cf i*))
⟨*proof*⟩

## 3.2 Quasi strong termination traces

**abbreviation** *qsend* $\equiv$ *sfirst* (*holds discrCf*)

**lemma** *qsend-eq-0-iff*: *qsend cfT* = *0* $\longleftrightarrow$ *discrCf* (*shd cfT*)
⟨*proof*⟩

**lemma** *qsend-eq-0*[*simp*]: *discrCf cf* $\Longrightarrow$ *qsend* (*cf* ## $\omega$) = *0*
⟨*proof*⟩

**lemma** *alw-discrCf*: *enabled cf* $\omega$ $\Longrightarrow$ *discrCf cf* $\Longrightarrow$ *alw* (*holds discrCf*) $\omega$
⟨*proof*⟩

**lemma** *alw-discrCf-indis*$'$:
*enabled cf* $\omega$ $\Longrightarrow$ *discrCf cf* $\Longrightarrow$ *snd cf* $\approx$ *t* $\Longrightarrow$ *alw* (*holds* ($\lambda$*cf*$'$. *snd cf*$'$ $\approx$ *t*)) $\omega$
⟨*proof*⟩

**lemma** *alw-discrCf-indis*:
*enabled cf* $\omega$ $\Longrightarrow$ *discrCf cf* $\Longrightarrow$ *alw* (*holds* ($\lambda$*cf*$'$. *snd cf*$'$ $\approx$ *snd cf*)) (*cf* ## $\omega$)
⟨*proof*⟩

**lemma** *enabled-sdrop*: *enabled cf* $\omega$ $\Longrightarrow$ *enabled* ((*cf* ## $\omega$) !! *n*) (*sdrop n* $\omega$)
⟨*proof*⟩

**lemma** *sfirst-eq-eSuc*: *sfirst P* $\omega$ = *eSuc n* $\longleftrightarrow$ ($\neg$ *P* $\omega$) $\wedge$ *sfirst P* (*stl* $\omega$) = *n*
⟨*proof*⟩

**lemma** *qsend-snth*: *qsend* $\omega$ = *enat n* $\Longrightarrow$ *discrCf* ($\omega$ !! *n*)
⟨*proof*⟩

**lemma** *indis-iff*: *a* $\approx$ *d* $\Longrightarrow$ *b* $\approx$ *d* $\Longrightarrow$ *a* $\approx$ *c* $\longleftrightarrow$ *b* $\approx$ *c*

⟨*proof*⟩

**lemma** *enabled-qsend-indis*:
**assumes** $enabled\ cf\ \omega\ qsend\ (cf\ \#\#\ \omega) \leq n\ qsend\ (cf\ \#\#\ \omega) \leq m$
**shows** $snd\ ((cf\ \#\#\ \omega)\ !!\ n) \approx t \longleftrightarrow snd\ ((cf\ \#\#\ \omega)\ !!\ m) \approx t$
⟨*proof*⟩

**lemma** *enabled-imp-UNTIL-alw-discrCf*:
$enabled\ (shd\ \omega)\ (stl\ \omega) \Longrightarrow (not\ (holds\ discrCf)\ until\ (alw\ (holds\ discrCf)))\ \omega$
⟨*proof*⟩

**lemma** *less-qsend-iff-not-discrCf*:
$enabled\ cf\ bT \Longrightarrow n < qsend\ (cf\ \#\#\ bT) \longleftrightarrow \neg\ discrCf\ ((cf\ \#\#\ bT)\ !!\ n)$
⟨*proof*⟩

## 3.3 Terminating configurations

**definition** $qstermCf\ cf \longleftrightarrow (\forall\, cfT.\ enabled\ cf\ cfT \longrightarrow qsend\ (cf\ \#\#\ cfT) < \infty)$

**lemma** *qstermCf-E*:
$qstermCf\ cf \Longrightarrow cf' \in G\ cf \Longrightarrow qstermCf\ cf'$
⟨*proof*⟩

**abbreviation** $eff\text{-}at\ cf\ bT\ n \equiv snd\ ((cf\ \#\#\ bT)\ !!\ n)$
**abbreviation** $cont\text{-}at\ cf\ bT\ n \equiv fst\ ((cf\ \#\#\ bT)\ !!\ n)$

**definition** $amSec\ c \longleftrightarrow (\forall\, s1\ s2\ n\ t.\ s1 \approx s2 \longrightarrow$
$\mathcal{P}(bT\ in\ T.T\ (c,\ s1).\ eff\text{-}at\ (c,\ s1)\ bT\ n \approx t) =$
$\mathcal{P}(bT\ in\ T.T\ (c,\ s2).\ eff\text{-}at\ (c,\ s2)\ bT\ n \approx t))$

**definition** $eSec\ c \longleftrightarrow (\forall\, s1\ s2\ t.\ s1 \approx s2 \longrightarrow$
$\mathcal{P}(bT\ in\ T.T\ (c,\ s1).\ \exists\, n.\ qsend\ ((c,\ s1)\ \#\#\ bT) = n \wedge eff\text{-}at\ (c,\ s1)\ bT\ n \approx$
$t) =$
$\mathcal{P}(bT\ in\ T.T\ (c,\ s2).\ \exists\, n.\ qsend\ ((c,\ s2)\ \#\#\ bT) = n \wedge eff\text{-}at\ (c,\ s2)\ bT\ n \approx$
$t))$

**definition** $aeT\ c \longleftrightarrow (\forall\, s.\ AE\ bT\ in\ T.T\ (c,s).\ qsend\ ((c,s)\ \#\#\ bT) < \infty)$

**lemma** *dist-Ps-upper-bound*:
**fixes** $cf1\ cf2 :: ('test,\ 'atom,\ 'choice)\ cmd \times 'state$ **and** $s :: 'state$ **and** $S$
**defines** $S\ cf\ bT \equiv \exists\, n.\ qsend\ (cf\ \#\#\ bT) = n \wedge eff\text{-}at\ cf\ bT\ n \approx s$
**defines** $Ps\ cf \equiv \mathcal{P}(bT\ in\ T.T\ cf.\ S\ cf\ bT)$
**defines** $N\ cf\ n\ bT \equiv \neg\ discrCf\ ((cf\ \#\#\ bT)\ !!\ n)$
**defines** $Pn\ cf\ n \equiv \mathcal{P}(bT\ in\ T.T\ cf.\ N\ cf\ n\ bT)$
**assumes** *bisim*: $proper\ (fst\ cf1)\ proper\ (fst\ cf2)\ fst\ cf1 \approx 01\ fst\ cf2\ snd\ cf1 \approx$
$snd\ cf2$
**shows** $dist\ (Ps\ cf1)\ (Ps\ cf2) \leq Pn\ cf1\ n + Pn\ cf2\ m$
⟨*proof*⟩

**lemma** *AE-T-max-qsend-time*:
**fixes** *cf* **and** *e* :: *real* **assumes** *AE*: *AE bT in T.T cf. qsend (cf ## bT) < ∞*
*0 < e*
**shows** ∃ *N*. 𝒫(*bT in T.T cf*. ¬ *discrCf* ((*cf* ## *bT*) !! *N*)) < *e*
⟨*proof*⟩

**lemma** *Ps-eq*:
**fixes** *cf1 cf2 s* **and** *S*
**defines** *S cf bT* ≡ ∃ *n*. *qsend* (*cf* ## *bT*) = *n* ∧ *eff-at cf bT n* ≈ *s*
**defines** *Ps cf* ≡ 𝒫(*bT in T.T cf*. *S cf bT*)
**assumes** *qsterm1*: *AE bT in T.T cf1*. *qsend* (*cf1* ## *bT*) < ∞
**assumes** *qsterm2*: *AE bT in T.T cf2*. *qsend* (*cf2* ## *bT*) < ∞
**and** *bisim*: *proper* (*fst cf1*) *proper* (*fst cf2*) *fst cf1* ≈*01 fst cf2 snd cf1* ≈ *snd cf2*
**shows** *Ps cf1* = *Ps cf2*
⟨*proof*⟩

**lemma** *siso-trace*:
**assumes** *siso c s* ≈ *t enabled* (*c*, *t*) *cfT*
**shows** *siso* (*cont-at* (*c*, *s*) *cfT n*)
**and** *cont-at* (*c*, *s*) *cfT n* = *cont-at* (*c*, *t*) *cfT n*
**and** *eff-at* (*c*, *s*) *cfT n* ≈ *eff-at* (*c*, *t*) *cfT n*
⟨*proof*⟩

**lemma** *Sbis-trace*:
**assumes** *proper* (*fst cf1*) *proper* (*fst cf2*) *fst cf1* ≈*s fst cf2 snd cf1* ≈ *snd cf2*
**shows** 𝒫(*cfT in T.T cf1*. *eff-at cf1 cfT n* ≈ *s'*) = 𝒫(*cfT in T.T cf2*. *eff-at cf2*
*cfT n* ≈ *s'*)
(**is** *?P cf1 n* = *?P cf2 n*)
⟨*proof*⟩

## 3.4 Final Theorems

**theorem** *ZObis-eSec*: ⟦*proper c*; *c* ≈*01 c*; *aeT c*⟧ ⟹ *eSec c*
⟨*proof*⟩

**theorem** *Sbis-amSec*: ⟦*proper c*; *c* ≈*s c*⟧ ⟹ *amSec c*
⟨*proof*⟩

**theorem** *amSec-eSec*:
**assumes** [*simp*]: *proper c* **and** *aeT c amSec c* **shows** *eSec c*
⟨*proof*⟩

**end**

**end**

# 4 Compositionality of Resumption-Based Noninterference

**theory** *Compositionality*
**imports** *Resumption-Based*
**begin**

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

## 4.1 Compatibility and discreetness of atoms, tests and choices

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

**definition** *presAtm* **where**
$presAtm\ atm \equiv$
$ALL\ s.\ s \approx aval\ atm\ s$

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

**lemma** *discrAt-compatAt*[*simp*]:
**assumes** *presAtm atm*
**shows** *compatAtm atm*
⟨*proof*⟩

**definition** *compatCh* **where**
$compatCh\ ch \equiv \forall\ s\ t.\ s \approx t \longrightarrow cval\ ch\ s = cval\ ch\ t$

**lemma** *compatCh-cval*[*simp*]:
**assumes** *compatCh ch* **and** $s \approx t$
**shows** $cval\ ch\ s = cval\ ch\ t$
⟨*proof*⟩

## 4.2 Compositionality of self-isomorphism

Self-Isomorphism versus language constructs:

**lemma** *siso-Done*[*simp*]:
*siso Done*
⟨*proof*⟩

**lemma** *siso-Atm*[*simp*]:
$siso\ (Atm\ atm) = compatAtm\ atm$
⟨*proof*⟩

**lemma** *siso-Seq*[*simp*]:
**assumes** ∗: *siso c1* **and** ∗∗: *siso c2*
**shows** *siso* (*c1* ;; *c2*)
⟨*proof*⟩

**lemma** *siso-While*[*simp*]:
**assumes** *compatTst tst* **and** *siso c*
**shows** *siso* (*While tst c*)
⟨*proof*⟩

**lemma** *siso-Ch*[*simp*]:
**assumes** *compatCh ch*
**and** ∗: *siso c1* **and** ∗∗: *siso c2*
**shows** *siso* (*Ch ch c1 c2*)
⟨*proof*⟩

**lemma** *siso-Par*[*simp*]:
**assumes** *properL cl* **and** *sisoL cl*
**shows** *siso* (*Par cl*)
⟨*proof*⟩

**lemma** *siso-ParT*[*simp*]:
**assumes** *properL cl* **and** *sisoL cl*
**shows** *siso* (*ParT cl*)
⟨*proof*⟩

Self-isomorphism implies strong bisimilarity:

**lemma** *bij-betw-emp*[*simp*]:
*bij-betw f* {} {}
⟨*proof*⟩

**lemma** *part-full*[*simp*]:
*part I* {*I*}
⟨*proof*⟩

**definition** *singlPart* **where**
*singlPart I* ≡ {{*i*} | *i* . *i* ∈ *I*}

**lemma** *part-singlPart*[*simp*]:
*part I* (*singlPart I*)
⟨*proof*⟩

**lemma** *singlPart-inj-on*[*simp*]:
*inj-on* (*image f*) (*singlPart I*) = *inj-on f I*
⟨*proof*⟩

**lemma** *singlPart-surj*[*simp*]:
(*image f*) ‘ (*singlPart I*) = (*singlPart J*) ⟷ *f* ‘ *I* = *J*

⟨*proof*⟩

**lemma** *singlPart-bij-betw*[*simp*]:
*bij-betw* (*image f*) (*singlPart I*) (*singlPart J*) = *bij-betw f I J*
⟨*proof*⟩

**lemma** *singlPart-finite1*:
**assumes** *finite* (*singlPart I*)
**shows** *finite* (*I*::′*a set*)
⟨*proof*⟩

**lemma** *singlPart-finite*[*simp*]:
*finite* (*singlPart I*) = *finite I*
⟨*proof*⟩

**lemma** *emp-notIn-singlPart*[*simp*]:
{} ∉ *singlPart I*
⟨*proof*⟩

**lemma** *Sbis-coinduct*[*consumes 1*, *case-names step*, *coinduct set*]:
*R c d* ⟹
(⋀*c d s t*. *R c d* ⟹ *s* ≈ *t* ⟹
∃ *P F*. *mC-C-part c d P F* ∧ *inj-on F P* ∧ *mC-C-wt c d s t P F* ∧
(∀ *I*∈*P*. ∀ *i*∈*I*. ∀ *j*∈*F I*.
*eff c s i* ≈ *eff d t j* ∧ (*R* (*cont c s i*) (*cont d t j*) ∨ (*cont c s i*, *cont d t j*) ∈ *Sbis*)))
⟹ (*c*, *d*) ∈ *Sbis*
⟨*proof*⟩

**lemma** *siso-Sbis*[*simp*]: *siso c* ⟹ *c* ≈*s c*
⟨*proof*⟩

## 4.3 Discreetness versus language constructs:

**lemma** *discr-Done*[*simp*]: *discr Done*
⟨*proof*⟩

**lemma** *discr-Atm-presAtm*[*simp*]: *discr* (*Atm atm*) = *presAtm atm*
⟨*proof*⟩

**lemma** *discr-Seq*[*simp*]:
*discr c1* ⟹ *discr c2* ⟹ *discr* (*c1* ;; *c2*)
⟨*proof*⟩

**lemma** *discr-While*[*simp*]: **assumes** *discr c* **shows** *discr* (*While tst c*)
⟨*proof*⟩

**lemma** *discr-Ch*[*simp*]: *discr c1* ⟹ *discr c2* ⟹ *discr* (*Ch ch c1 c2*)
⟨*proof*⟩

**lemma** *discr-Par*[*simp*]: *properL cl* $\Longrightarrow$ *discrL cl* $\Longrightarrow$ *discr* (*Par cl*)
⟨*proof*⟩

**lemma** *discr-ParT*[*simp*]: *properL cl* $\Longrightarrow$ *discrL cl* $\Longrightarrow$ *discr* (*ParT cl*)
⟨*proof*⟩

**lemma** *discr-finished*[*simp*]: *proper c* $\Longrightarrow$ *finished c* $\Longrightarrow$ *discr c*
⟨*proof*⟩

## 4.4 Strong bisimilarity versus language constructs

**lemma** *Sbis-pres-discr-L*:
*c* $\approx$*s d* $\Longrightarrow$ *discr d* $\Longrightarrow$ *discr c*
⟨*proof*⟩

**lemma** *Sbis-pres-discr-R*:
**assumes** *discr c* **and** *c* $\approx$*s d*
**shows** *discr d*
⟨*proof*⟩

**lemma** *Sbis-finished-discr-L*:
**assumes** *c* $\approx$*s d* **and** *proper d* **and** *finished d*
**shows** *discr c*
⟨*proof*⟩

**lemma** *Sbis-finished-discr-R*:
**assumes** *proper c* **and** *finished c* **and** *c* $\approx$*s d*
**shows** *discr d*
⟨*proof*⟩

**definition** *thetaSD* **where**
*thetaSD* $\equiv$ {(*c*, *d*) | *c d* . *proper c* $\wedge$ *proper d* $\wedge$ *discr c* $\wedge$ *discr d*}

**lemma** *thetaSD-Sretr*:
*thetaSD* $\subseteq$ *Sretr thetaSD*
⟨*proof*⟩

**lemma** *thetaSD-Sbis*:
*thetaSD* $\subseteq$ *Sbis*
⟨*proof*⟩

**theorem** *discr-Sbis*[*simp*]:
**assumes** *proper c* **and** *proper d* **and** *discr c* **and** *discr d*
**shows** *c* $\approx$*s d*
⟨*proof*⟩

**definition** *thetaSDone* **where**
*thetaSDone* ≡ {(*Done*, *Done*)}

**lemma** *thetaSDone-Sretr*:
*thetaSDone* ⊆ *Sretr thetaSDone*
⟨*proof*⟩

**lemma** *thetaSDone-Sbis*:
*thetaSDone* ⊆ *Sbis*
⟨*proof*⟩

**theorem** *Done-Sbis*[*simp*]:
*Done* ≈*s Done*
⟨*proof*⟩

**definition** *thetaSAtm* **where**
*thetaSAtm atm* ≡
{(*Atm atm*, *Atm atm*), (*Done*, *Done*)}

**lemma** *thetaSAtm-Sretr*:
**assumes** *compatAtm atm*
**shows** *thetaSAtm atm* ⊆ *Sretr* (*thetaSAtm atm*)
⟨*proof*⟩

**lemma** *thetaSAtm-Sbis*:
**assumes** *compatAtm atm*
**shows** *thetaSAtm atm* ⊆ *Sbis*
⟨*proof*⟩

**theorem** *Atm-Sbis*[*simp*]:
**assumes** *compatAtm atm*
**shows** *Atm atm* ≈*s Atm atm*
⟨*proof*⟩

**definition** *thetaSSeqI* **where**
*thetaSSeqI* ≡
{(*e* ;; *c*, *e* ;; *d*) | *e c d* . *siso e* ∧ *c* ≈*s d*}

**lemma** *thetaSSeqI-Sretr*:
*thetaSSeqI* ⊆ *Sretr* (*thetaSSeqI Un Sbis*)
⟨*proof*⟩

**lemma** *thetaSSeqI-Sbis*:
*thetaSSeqI* ⊆ *Sbis*
⟨*proof*⟩

**theorem** *Seq-siso-Sbis*[*simp*]:
**assumes** *siso e* **and** *c2* ≈*s d2*
**shows** *e* ;; *c2* ≈*s e* ;; *d2*
⟨*proof*⟩

**definition** *thetaSSeqD* **where**
*thetaSSeqD* ≡
{(*c1* ;; *c2*, *d1* ;; *d2*) |
  *c1 c2 d1 d2*.
    *proper c1* ∧ *proper d1* ∧ *proper c2* ∧ *proper d2* ∧
    *discr c2* ∧ *discr d2* ∧
    *c1* ≈*s d1*}

**lemma** *thetaSSeqD-Sretr*:
*thetaSSeqD* ⊆ *Sretr* (*thetaSSeqD Un Sbis*)
⟨*proof*⟩

**lemma** *thetaSSeqD-Sbis*:
*thetaSSeqD* ⊆ *Sbis*
⟨*proof*⟩

**theorem** *Seq-Sbis*[*simp*]:
**assumes** *proper c1* **and** *proper d1* **and** *proper c2* **and** *proper d2*
**and** *c1* ≈*s d1* **and** *discr c2* **and** *discr d2*
**shows** *c1* ;; *c2* ≈*s d1* ;; *d2*
⟨*proof*⟩

**definition** *thetaSCh* **where**
*thetaSCh ch c1 c2 d1 d2* ≡ {(*Ch ch c1 c2*, *Ch ch d1 d2*)}

**lemma** *thetaSCh-Sretr*:
**assumes** *compatCh ch* **and** *c1* ≈*s d1* **and** *c2* ≈*s d2*
**shows** *thetaSCh ch c1 c2 d1 d2* ⊆
  *Sretr* (*thetaSCh ch c1 c2 d1 d2* ∪ *Sbis*)
(**is** *?th* ⊆ *Sretr* (*?th* ∪ *Sbis*))
⟨*proof*⟩

**lemma** *thetaSCh-Sbis*:
**assumes** *compatCh ch* **and** *c1* ≈*s d1* **and** *c2* ≈*s d2*
**shows** *thetaSCh ch c1 c2 d1 d2* ⊆ *Sbis*
⟨*proof*⟩

**theorem** *Ch-siso-Sbis*[*simp*]:
**assumes** *compatCh ch* **and** *c1* ≈*s d1* **and** *c2* ≈*s d2*

**shows** *Ch ch c1 c2* ≈*s Ch ch d1 d2*
⟨*proof*⟩

**definition** *shift* **where**
*shift cl n* ≡ *image* (%*i*. *brnL cl n* + *i*)

**definition** *back* **where**
*back cl n* ≡ *image* (% *ii*. *ii* − *brnL cl n*)

**lemma** *emp-shift*[*simp*]:
*shift cl n I* = {} ⟷ *I* = {}
⟨*proof*⟩

**lemma** *emp-shift-rev*[*simp*]:
{} = *shift cl n I* ⟷ *I* = {}
⟨*proof*⟩

**lemma** *emp-back*[*simp*]:
*back cl n II* = {} ⟷ *II* = {}
⟨*proof*⟩

**lemma** *emp-back-rev*[*simp*]:
{} = *back cl n II* ⟷ *II* = {}
⟨*proof*⟩

**lemma** *in-shift*[*simp*]:
*brnL cl n* + *i* ∈ *shift cl n I* ⟷ *i* ∈ *I*
⟨*proof*⟩

**lemma** *in-back*[*simp*]:
*ii* ∈ *II* ⟹ *ii* − *brnL cl n* ∈ *back cl n II*
⟨*proof*⟩

**lemma** *in-back2*[*simp*]:
**assumes** *ii* > *brnL cl n* **and** *II* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)}
**shows** *ii* − *brnL cl n* ∈ *back cl n II* ⟷ *ii* ∈ *II* (**is** *?L* ⟷ *?R*)
⟨*proof*⟩

**lemma** *shift*[*simp*]:
**assumes** *I* ⊆ {..< *brn* (*cl*!*n*)}
**shows** *shift cl n I* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)}
⟨*proof*⟩

**lemma** *shift2*[*simp*]:
**assumes** *I* ⊆ {..< *brn* (*cl*!*n*)}
**and** *ii* ∈ *shift cl n I*
**shows** *brnL cl n* ≤ *ii* ∧ *ii* < *brnL cl n* + *brn* (*cl*!*n*)

⟨*proof*⟩

**lemma** *shift3*[*simp*]:
**assumes** *n*: $n < length\ cl$ **and** *I*: $I \subseteq \{..< brn\ (cl!n)\}$
**and** *ii*: $ii \in shift\ cl\ n\ I$
**shows** $ii < brnL\ cl\ (length\ cl)$
⟨*proof*⟩

**lemma** *back*[*simp*]:
**assumes** $II \subseteq \{brnL\ cl\ n\ ..<+\ brn\ (cl!n)\}$
**shows** $back\ cl\ n\ II \subseteq \{..< brn\ (cl!n)\}$
⟨*proof*⟩

**lemma** *back2*[*simp*]:
**assumes** $II \subseteq \{brnL\ cl\ n\ ..<+\ brn\ (cl!n)\}$
**and** $i \in back\ cl\ n\ II$
**shows** $i < brn\ (cl!n)$
⟨*proof*⟩

**lemma** *shift-inj*[*simp*]:
$shift\ cl\ n\ I1 = shift\ cl\ n\ I2 \longleftrightarrow I1 = I2$
⟨*proof*⟩

**lemma** *shift-mono*[*simp*]:
$shift\ cl\ n\ I1 \subseteq shift\ cl\ n\ I2 \longleftrightarrow I1 \subseteq I2$
⟨*proof*⟩

**lemma** *shift-Int*[*simp*]:
$shift\ cl\ n\ I1 \cap shift\ cl\ n\ I2 = \{\} \longleftrightarrow I1 \cap I2 = \{\}$
⟨*proof*⟩

**lemma** *inj-shift*: $inj\ (shift\ cl\ n)$
⟨*proof*⟩

**lemma** *inj-on-shift*: $inj\text{-}on\ (shift\ cl\ n)\ K$
⟨*proof*⟩

**lemma** *back-shift*[*simp*]:
$back\ cl\ n\ (shift\ cl\ n\ I) = I$
⟨*proof*⟩

**lemma** *shift-back*[*simp*]:
**assumes** $II \subseteq \{brnL\ cl\ n\ ..<+\ brn\ (cl!n)\}$
**shows** $shift\ cl\ n\ (back\ cl\ n\ II) = II$
⟨*proof*⟩

**lemma** *back-inj*[*simp*]:
**assumes** *II1*: $II1 \subseteq \{brnL\ cl\ n\ ..<+\ brn\ (cl!n)\}$
**and** *II2*: $II2 \subseteq \{brnL\ cl\ n\ ..<+\ brn\ (cl!n)\}$

**shows** *back cl n II1* = *back cl n II2* ⟷ *II1* = *II2* (**is** *?L* = *?R* ⟷ *II1* = *II2*)
⟨*proof*⟩

**lemma** *back-mono*[*simp*]:
**assumes** *II1* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)}
**and** *II2* ⊆ {*brnL cl n* ..< *brnL cl n* + *brn* (*cl*!*n*)}
**shows** *back cl n II1* ⊆ *back cl n II2* ⟷ *II1* ⊆ *II2*
(**is** *?L* ⊆ *?R* ⟷ *II1* ⊆ *II2*)
⟨*proof*⟩

**lemma** *back-Int*[*simp*]:
**assumes** *II1* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)}
**and** *II2* ⊆ {*brnL cl n* ..< *brnL cl n* + *brn* (*cl*!*n*)}
**shows** *back cl n II1* ∩ *back cl n II2* = {} ⟷ *II1* ∩ *II2* = {}
(**is** *?L* ∩ *?R* = {} ⟷ *II1* ∩ *II2* = {})
⟨*proof*⟩

**lemma** *inj-on-back*:
*inj-on* (*back cl n*) (*Pow* {*brnL cl n* ..<+ *brn* (*cl*!*n*)})
⟨*proof*⟩

**lemma** *shift-surj*:
**assumes** *II* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)}
**shows** ∃ *I*. *I* ⊆ {..< *brn* (*cl*!*n*)} ∧ *shift cl n I* = *II*
⟨*proof*⟩

**lemma** *back-surj*:
**assumes** *I* ⊆ {..< *brn* (*cl*!*n*)}
**shows** ∃ *II*. *II* ⊆ {*brnL cl n* ..<+ *brn* (*cl*!*n*)} ∧ *back cl n II* = *I*
⟨*proof*⟩

**lemma** *shift-part*[*simp*]:
**assumes** *part* {..< *brn* (*cl*!*n*)} *P*
**shows** *part* {*brnL cl n* ..<+ *brn* (*cl*!*n*)} (*shift cl n* ‘ *P*)
⟨*proof*⟩

**lemma** *part-brn-disj1*:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*)
**and** *n1*: *n1* < *length cl* **and** *n2*: *n2* < *length cl*
**and** *II1*: *II1* ∈ *shift cl n1* ‘ (*P n1*) **and** *II2*: *II2* ∈ *shift cl n2* ‘ (*P n2*) **and** *d*: *n1* ≠ *n2*
**shows** *II1* ∩ *II2* = {}
⟨*proof*⟩

**lemma** *part-brn-disj2*:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *n1*: *n1* < *length cl* **and** *n2*: *n2* < *length cl* **and** *d*: *n1* ≠ *n2*
**shows** *shift cl n1* ‘ (*P n1*) ∩ *shift cl n2* ‘ (*P n2*) = {} (**is** *?L* ∩ *?R* = {})
⟨*proof*⟩

**lemma** *part-brn-disj3*:
**assumes** *P*: $\bigwedge$ *n. n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*)
**and** *n1*: *n1* < *length cl* **and** *n2*: *n2* < *length cl*
**and** *I1*: *I1* ∈ *P n1* **and** *I2*: *I2* ∈ *P n2* **and** *d*: *n1* ≠ *n2*
**shows** *shift cl n1 I1* ∩ *shift cl n2 I2* = {}
⟨*proof*⟩

**lemma** *sum-wt-Par-sub-shift*[*simp*]:
**assumes** *cl*: *properL cl* **and** *n*: *n* < *length cl* **and**
*I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows**
*sum* (*wt* (*Par cl*) *s*) (*shift cl n I*) =
*1* / (*length cl*) ∗ *sum* (*wt* (*cl* ! *n*) *s*) *I*
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-WtFT-pickFT-0-shift*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1*
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! (*pickFT cl*))} *0* ∈ *I*
**shows**
*sum* (*wt* (*ParT cl*) *s*) (*shift cl* (*pickFT cl*) *I*) = *1*
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-WtFT-notPickFT-0-shift*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* = *1* **and** *n*: *n* < *length cl*
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)} **and** *nI*: *n* = *pickFT cl* ⟶ *0* ∉ *I*
**shows** *sum* (*wt* (*ParT cl*) *s*) (*shift cl n I*) = *0*
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-notWtFT-finished-shift*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* ≠ *1* **and** *n*: *n* < *length cl* **and** *cln*: *finished* (*cl*!*n*)
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows** *sum* (*wt* (*ParT cl*) *s*) (*shift cl n I*) = *0*
⟨*proof*⟩

**lemma** *sum-wt-ParT-sub-notWtFT-notFinished-shift*[*simp*]:
**assumes** *cl*: *properL cl* **and** *nf*: *WtFT cl* ≠ *1*
**and** *n*: *n* < *length cl* **and** *cln*: ¬ *finished* (*cl*!*n*)
**and** *I*: *I* ⊆ {..< *brn* (*cl* ! *n*)}
**shows**
*sum* (*wt* (*ParT cl*) *s*) (*shift cl n I*) =
(*1* / (*length cl*)) / (*1* − *WtFT cl*) ∗ *sum* (*wt* (*cl* ! *n*) *s*) *I*
⟨*proof*⟩

**definition** *UNpart* **where**
*UNpart cl P* ≡ $\bigcup$ *n* < *length cl*. *shift cl n* ‘ (*P n*)

**lemma** *UNpart-cases*[*elim*, *consumes 1*, *case-names Local*]:
**assumes** *II* ∈ *UNpart cl P* **and**
⋀ *n I*. ⟦*n* < *length cl*; *I* ∈ *P n*; *II* = *shift cl n I*⟧ ⟹ *phi*
**shows** *phi*
⟨*proof*⟩

**lemma** *emp-UNpart*:
**assumes** ⋀ *n*. *n* < *length cl* ⟹ {} ∉ *P n*
**shows** {} ∉ *UNpart cl P*
⟨*proof*⟩

**lemma** *part-UNpart*:
**assumes** *cl*: *properL cl* **and**
*P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*)
**shows** *part* {..< *brnL cl* (*length cl*)} (*UNpart cl P*)
(**is** *part ?J ?Q*)
⟨*proof*⟩

**definition** *pickT-pred* **where**
*pickT-pred cl P II n* ≡ *n* < *length cl* ∧ *II* ∈ *shift cl n* ' (*P n*)

**definition** *pickT* **where**
*pickT cl P II* ≡ *SOME n*. *pickT-pred cl P II n*

**lemma** *pickT-pred*:
**assumes** *II* ∈ *UNpart cl P*
**shows** ∃ *n*. *pickT-pred cl P II n*
⟨*proof*⟩

**lemma** *pickT-pred-unique*:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *1*: *pickT-pred cl P II n1* **and** *2*: *pickT-pred cl P II n2*
**shows** *n1* = *n2*
⟨*proof*⟩

**lemma** *pickT-pred-pickT*:
**assumes** *II* ∈ *UNpart cl P*
**shows** *pickT-pred cl P II* (*pickT cl P II*)
⟨*proof*⟩

**lemma** *pickT-pred-pickT-unique*:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *pickT-pred cl P II n*
**shows** *n* = *pickT cl P II*
⟨*proof*⟩

**lemma** *pickT-length*[*simp*]:
**assumes** *II* ∈ *UNpart cl P*
**shows** *pickT cl P II* < *length cl*
⟨*proof*⟩

**lemma** *pickT-shift*[*simp*]:
**assumes** *II* ∈ *UNpart cl P*
**shows** *II* ∈ *shift cl* (*pickT cl P II*) ' (*P* (*pickT cl P II*))
⟨*proof*⟩

**lemma** *pickT-unique*:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *n* < *length cl* **and** *II* ∈ *shift cl n* ' (*P n*)
**shows** *n* = *pickT cl P II*
⟨*proof*⟩

**definition** *UNlift* **where**
*UNlift cl dl P F II* ≡
*shift dl* (*pickT cl P II*) (*F* (*pickT cl P II*) (*back cl* (*pickT cl P II*) *II*))

**lemma** *UNlift-shift*[*simp*]:
**assumes** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *n*: *n* < *length cl* **and** *I*: *I* ∈ *P n*
**shows** *UNlift cl dl P F* (*shift cl n I*) = *shift dl n* (*F n I*)
⟨*proof*⟩

**lemma** *UNlift-inj-on*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: ⋀ *n*. *n* < *length dl* ⟹ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*)) ∧ {} ∉ *F n* ' (*P n*)
**and** *F*: ⋀ *n*. *n* < *length cl* ⟹ *inj-on* (*F n*) (*P n*)
**shows** *inj-on* (*UNlift cl dl P F*) (*UNpart cl P*) (**is** *inj-on ?G ?Q*)
⟨*proof*⟩

**lemma** *UNlift-UNpart*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** (*UNlift cl dl P F*) ' (*UNpart cl P*) = *UNpart dl* (%*n*. *F n* ' (*P n*)) (**is** *?G* ' *?Q* = *?R*)
⟨*proof*⟩

**lemma** *emp-UNlift-UNpart*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: ⋀ *n*. *n* < *length dl* ⟹ {} ∉ *F n* ' (*P n*)
**shows** {} ∉ (*UNlift cl dl P F*) ' (*UNpart cl P*) (**is** {} ∉ *?R*)
⟨*proof*⟩

**lemma** *part-UNlift-UNpart*:
**assumes** *l*: *length cl* = *length dl* **and** *dl*: *properL dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* ⟹ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*))
**shows** *part* {..< *brnL dl* (*length dl*)} ((*UNlift cl dl P F*) ' (*UNpart cl P*)) (**is** *part ?C ?R*)
⟨*proof*⟩

**lemma** *ss-wt-Par-UNlift*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl* **and** *II*: *II* ∈ *UNpart cl P*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* ⟹ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*))
**and** *sw*:
$\bigwedge$*n I*. ⟦*n* < *length cl*; *I* ∈ *P n*⟧ ⟹
    *sum* (*wt* (*cl* ! *n*) *s*) *I* =
    *sum* (*wt* (*dl* ! *n*) *t*) (*F n I*)
**and** *st*: *s* ≈ *t*
**shows**
*sum* (*wt* (*Par cl*) *s*) *II* =
*sum* (*wt* (*Par dl*) *t*) (*UNlift cl dl P F II*) (**is** *?L* = *?R*)
⟨*proof*⟩

**definition** *thetaSPar* **where**
*thetaSPar* ≡
{(*Par cl*, *Par dl*) |
  *cl dl*. *properL cl* ∧ *properL dl* ∧ *SbisL cl dl*}

**lemma** *cont-eff-Par-UNlift*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl SbisL cl dl*
**and** *II*: *II* ∈ *UNpart cl P* **and** *ii*: *ii* ∈ *II* **and** *jj*: *jj* ∈ *UNlift cl dl P F II*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* ⟹ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*))
**and** *eff-cont*:
$\bigwedge$*n I i j*. ⟦*n* < *length cl*; *I* ∈ *P n*; *i* ∈ *I*; *j* ∈ *F n I*⟧ ⟹
  *eff* (*cl*!*n*) *s i* ≈ *eff* (*dl*!*n*) *t j* ∧
  *cont* (*cl*!*n*) *s i* ≈*s* *cont* (*dl*!*n*) *t j*
**and** *st*: *s* ≈ *t*
**shows**
*eff* (*Par cl*) *s ii* ≈ *eff* (*Par dl*) *t jj* ∧
(*cont* (*Par cl*) *s ii*, *cont* (*Par dl*) *t jj*) ∈ *thetaSPar*
(**is** *?eff* ∧ *?cont*)
⟨*proof*⟩

**lemma** *thetaSPar-Sretr*: *thetaSPar* ⊆ *Sretr* (*thetaSPar*)
⟨*proof*⟩

**lemma** *thetaSPar-Sbis*: *thetaSPar* $\subseteq$ *Sbis*
⟨*proof*⟩

**theorem** *Par-Sbis*[*simp*]:
**assumes** *properL cl* **and** *properL dl SbisL cl dl*
**shows** *Par cl* $\approx$*s Par dl*
⟨*proof*⟩

## 4.5 01-bisimilarity versus language constructs

**lemma** *ZObis-pres-discr-L*: *c* $\approx$*01 d* $\Longrightarrow$ *discr d* $\Longrightarrow$ *discr c*
⟨*proof*⟩

**theorem** *ZObis-pres-discr-R*:
**assumes** *discr c* **and** *c* $\approx$*01 d*
**shows** *discr d*
⟨*proof*⟩

**theorem** *ZObis-finished-discr-L*:
**assumes** *c* $\approx$*01 d* **and** *proper d* **and** *finished d*
**shows** *discr c*
⟨*proof*⟩

**theorem** *ZObis-finished-discr-R*:
**assumes** *proper c* **and** *finished c* **and** *c* $\approx$*01 d*
**shows** *discr d*
⟨*proof*⟩

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

**theorem** *Done-ZObis*[*simp*]:
*Done* $\approx$*01 Done*
⟨*proof*⟩

**theorem** *Atm-ZObis*[*simp*]:
**assumes** *compatAtm atm*
**shows** *Atm atm* $\approx$*01 Atm atm*
⟨*proof*⟩

**definition** *thetaZOSeqI* **where**
*thetaZOSeqI* $\equiv$

$\{(e\ ;;\ c,\ e\ ;;\ d) \mid e\ c\ d\ .\ siso\ e \wedge c \approx 01\ d\}$

**lemma** *thetaZOSeqI-ZOretr*:
*thetaZOSeqI* ⊆ *ZOretr* (*thetaZOSeqI Un ZObis*)
⟨*proof*⟩

**lemma** *thetaZOSeqI-ZObis*:
*thetaZOSeqI* ⊆ *ZObis*
⟨*proof*⟩

**theorem** *Seq-siso-ZObis*[*simp*]:
**assumes** *siso e* **and** *c2* ≈*01 d2*
**shows** *e* ;; *c2* ≈*01 e* ;; *d2*
⟨*proof*⟩

**definition** *thetaZOSeqD* **where**
*thetaZOSeqD* ≡
{(*c1* ;; *c2*, *d1* ;; *d2*) |
*c1 c2 d1 d2*.
*proper c1* ∧ *proper d1* ∧ *proper c2* ∧ *proper d2* ∧
*discr c2* ∧ *discr d2* ∧
*c1* ≈*01 d1*}

**lemma** *thetaZOSeqD-ZOretr*:
*thetaZOSeqD* ⊆ *ZOretr* (*thetaZOSeqD Un ZObis*)
⟨*proof*⟩

**lemma** *thetaZOSeqD-ZObis*:
*thetaZOSeqD* ⊆ *ZObis*
⟨*proof*⟩

**theorem** *Seq-ZObis*[*simp*]:
**assumes** *proper c1* **and** *proper d1* **and** *proper c2* **and** *proper d2*
**and** *c1* ≈*01 d1* **and** *discr c2* **and** *discr d2*
**shows** *c1* ;; *c2* ≈*01 d1* ;; *d2*
⟨*proof*⟩

**definition** *thetaZOCh* **where**
*thetaZOCh ch c1 c2 d1 d2* ≡ {(*Ch ch c1 c2*, *Ch ch d1 d2*)}

**lemma** *thetaZOCh-Sretr*:
**assumes** *compatCh ch* **and** *c1* ≈*01 d1* **and** *c2* ≈*01 d2*
**shows** *thetaZOCh ch c1 c2 d1 d2* ⊆
*Sretr* (*thetaZOCh ch c1 c2 d1 d2* ∪ *ZObis*)
(**is** *?th* ⊆ *Sretr* (*?th* ∪ *ZObis*))
⟨*proof*⟩

**lemma** *thetaZOCh-ZOretr*:
**assumes** *compatCh ch* **and** $c1 \approx 01\ d1$ **and** $c2 \approx 01\ d2$
**shows** *thetaZOCh ch c1 c2 d1 d2* $\subseteq$
*ZOretr* (*thetaZOCh ch c1 c2 d1 d2* $\cup$ *ZObis*)
⟨*proof*⟩

**lemma** *thetaZOCh-ZObis*:
**assumes** *compatCh ch* **and** $c1 \approx 01\ d1$ **and** $c2 \approx 01\ d2$
**shows** *thetaZOCh ch c1 c2 d1 d2* $\subseteq$ *ZObis*
⟨*proof*⟩

**theorem** *Ch-siso-ZObis*[*simp*]:
**assumes** *compatCh ch* **and** $c1 \approx 01\ d1$ **and** $c2 \approx 01\ d2$
**shows** *Ch ch c1 c2* $\approx 01$ *Ch ch d1 d2*
⟨*proof*⟩

**definition** *theFTOne* **where**
*theFTOne cl dl* $\equiv$ *theFT cl* $\cup$ *theFT dl*

**definition** *theNFTBoth* **where**
*theNFTBoth cl dl* $\equiv$ *theNFT cl* $\cap$ *theNFT dl*

**lemma** *theFTOne-sym*: *theFTOne cl dl* = *theFTOne dl cl*
⟨*proof*⟩

**lemma** *finite-theFTOne*[*simp*]:
*finite* (*theFTOne cl dl*)
⟨*proof*⟩

**lemma** *theFTOne-length-finished*[*simp*]:
**assumes** $n \in$ *theFTOne cl dl*
**shows** ($n <$ *length cl* $\wedge$ *finished* (*cl*!*n*)) $\vee$ ($n <$ *length dl* $\wedge$ *finished* (*dl*!*n*))
⟨*proof*⟩

**lemma** *theFTOne-length*[*simp*]:
**assumes** *length cl* = *length dl* **and** $n \in$ *theFTOne cl dl*
**shows** $n <$ *length cl* **and** $n <$ *length dl*
⟨*proof*⟩

**lemma** *theFTOne-intro*[*intro*]:
**assumes** $\bigwedge n.$ ($n <$ *length cl* $\wedge$ *finished* (*cl*!*n*)) $\vee$ ($n <$ *length dl* $\wedge$ *finished* (*dl*!*n*))
**shows** $n \in$ *theFTOne cl dl*
⟨*proof*⟩

**lemma** *pickFT-theFTOne*[*simp*]:
**assumes** *WtFT cl* = *1*
**shows** *pickFT cl* $\in$ *theFTOne cl dl*

⟨*proof*⟩

**lemma** *finite-theNFTBoth*[*simp*]:
*finite* (*theNFTBoth cl dl*)
⟨*proof*⟩

**lemma** *theNFTBoth-sym*: *theNFTBoth cl dl* = *theNFTBoth dl cl*
⟨*proof*⟩

**lemma** *theNFTBoth-length-finished*[*simp*]:
**assumes** $n \in$ *theNFTBoth cl dl*
**shows** $n <$ *length cl* **and** $\neg$ *finished* (*cl*!*n*)
**and** $n <$ *length dl* **and** $\neg$ *finished* (*dl*!*n*)
⟨*proof*⟩

**lemma** *theNFTBoth-intro*[*intro*]:
**assumes** $\bigwedge$ *n*. $n <$ *length cl* $\wedge \neg$ *finished* (*cl*!*n*) $\wedge$ $n <$ *length dl* $\wedge \neg$ *finished* (*dl*!*n*)
**shows** $n \in$ *theNFTBoth cl dl*
⟨*proof*⟩

**lemma** *theFTOne-Int-theNFTBoth*[*simp*]:
*theFTOne cl dl* $\cap$ *theNFTBoth cl dl* = {}
**and** *theNFTBoth cl dl* $\cap$ *theFTOne cl dl* = {}
⟨*proof*⟩

**lemma** *theFT-Un-theNFT-One-Both*[*simp*]:
**assumes** *length cl* = *length dl*
**shows**
*theFTOne cl dl* $\cup$ *theNFTBoth cl dl* = {..< *length cl*} **and**
*theNFTBoth cl dl* $\cup$ *theFTOne cl dl* = {..< *length cl*}
⟨*proof*⟩

**lemma** *in-theFTOne-theNFTBoth*[*simp*]:
**assumes** *n1* $\in$ *theFTOne cl dl* **and** *n2* $\in$ *theNFTBoth cl dl*
**shows** *n1* $\neq$ *n2* **and** *n2* $\neq$ *n1*
⟨*proof*⟩

**definition** *BrnFT* **where**
*BrnFT cl dl* $\equiv \bigcup$ $n \in$ *theFTOne cl dl*. {*brnL cl n* ..<+ *brn* (*cl*!*n*)}

**definition** *BrnNFT* **where**
*BrnNFT cl dl* $\equiv \bigcup$ $n \in$ *theNFTBoth cl dl*. {*brnL cl n* ..<+ *brn* (*cl*!*n*)}

**lemma** *BrnFT-elim*[*elim*, *consumes 1*, *case-names Local*]:
**assumes** *ii* $\in$ *BrnFT cl dl*

**and** $\bigwedge$ *n i*. $[\![n \in theFTOne\ cl\ dl;\ i < brn\ (cl!n);\ ii = brnL\ cl\ n + i]\!] \Longrightarrow phi$
**shows** *phi*
⟨*proof*⟩

**lemma** *finite-BrnFT*[*simp*]:
*finite* (*BrnFT cl dl*)
⟨*proof*⟩

**lemma** *BrnFT-incl-brnL*[*simp*]:
**assumes** *l*: *length cl* = *length dl* **and** *cl*: *properL cl*
**shows** *BrnFT cl dl* ⊆ {..< *brnL cl* (*length cl*)} (**is** *?L* ⊆ *?R*)
⟨*proof*⟩

**lemma** *BrnNFT-elim*[*elim*, *consumes 1*, *case-names Local*]:
**assumes** *ii* ∈ *BrnNFT cl dl*
**and** $\bigwedge$ *n i*. $[\![n \in theNFTBoth\ cl\ dl;\ i < brn\ (cl!n);\ ii = brnL\ cl\ n + i]\!] \Longrightarrow phi$
**shows** *phi*
⟨*proof*⟩

**lemma** *finite-BrnNFT*[*simp*]:
*finite* (*BrnNFT cl dl*)
⟨*proof*⟩

**lemma** *BrnNFT-incl-brnL*[*simp*]:
**assumes** *cl*: *properL cl*
**shows** *BrnNFT cl dl* ⊆ {..< *brnL cl* (*length cl*)} (**is** *?L* ⊆ *?R*)
⟨*proof*⟩

**lemma** *BrnFT-Int-BrnNFT*[*simp*]:
**assumes** *l*: *length cl* = *length dl*
**shows**
*BrnFT cl dl* ∩ *BrnNFT cl dl* = {} (**is** *?L*)
**and** *BrnNFT cl dl* ∩ *BrnFT cl dl* = {} (**is** *?R*)
⟨*proof*⟩

**lemma** *BrnFT-Un-BrnNFT*[*simp*]:
**assumes** *l*: *length cl* = *length dl* **and** *cl*: *properL cl*
**shows** *BrnFT cl dl* ∪ *BrnNFT cl dl* = {..< *brnL cl* (*length cl*)} (**is** *?L1* = *?R*)
**and** *BrnNFT cl dl* ∪ *BrnFT cl dl* = {..< *brnL cl* (*length cl*)} (**is** *?L2* = *?R*)
⟨*proof*⟩

**lemma** *BrnFT-part*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*)
**shows** *BrnFT cl dl* = ($\bigcup$ *n* ∈ *theFTOne cl dl*. *Union* (*shift cl n* ‘ (*P n*))) (**is** *?L* = *?R*)
⟨*proof*⟩

**lemma** *brnL-pickFT-BrnFT*[*simp*]:

**assumes** *properL cl* **and** *WtFT cl = 1*
**shows** *brnL cl (pickFT cl) ∈ BrnFT cl dl*
⟨*proof*⟩

**lemma** *WtFT-ParT-BrnFT*[*simp*]:
**assumes** *length cl = length dl properL cl* **and** *WtFT cl = 1*
**shows** *sum (wt (ParT cl) s) (BrnFT cl dl) = 1*
⟨*proof*⟩

**definition** *UNpart1* **where**
*UNpart1 cl dl P ≡ ⋃ n ∈ theNFTBoth cl dl. shift cl n ' (P n)*

**definition** *UNpart01* **where**
*UNpart01 cl dl P ≡ {BrnFT cl dl} ∪ UNpart1 cl dl P*

**lemma** *BrnFT-UNpart01*[*simp*]:
*BrnFT cl dl ∈ UNpart01 cl dl P*
⟨*proof*⟩

**lemma** *UNpart1-cases*[*elim, consumes 1, case-names Local*]:
**assumes** *II ∈ UNpart1 cl dl P*
*⋀ n I. ⟦n ∈ theNFTBoth cl dl; I ∈ P n; II = shift cl n I⟧ ⟹ phi*
**shows** *phi*
⟨*proof*⟩

**lemma** *UNpart01-cases*[*elim, consumes 1, case-names Local0 Local*]:
**assumes** *II ∈ UNpart01 cl dl P* **and** *II = BrnFT cl dl ⟹ phi*
*⋀ n I. ⟦n ∈ theNFTBoth cl dl; I ∈ P n; II = shift cl n I; II ∈ UNpart1 cl dl P⟧*
*⟹ phi*
**shows** *phi*
⟨*proof*⟩

**lemma** *emp-UNpart1*:
**assumes** *⋀ n. n < length cl ⟹ {} ∉ P n*
**shows** *{} ∉ UNpart1 cl dl P*
⟨*proof*⟩

**lemma** *emp-UNpart01*:
**assumes** *⋀ n. n < length cl ⟹ {} ∉ P n*
**shows** *{} ∉ UNpart01 cl dl P − {BrnFT cl dl}*
⟨*proof*⟩

**lemma** *BrnFT-Int-UNpart1*[*simp*]:
**assumes** *l*: *length cl = length dl*
**and** *P*: *⋀ n. n < length cl ⟹ part {..< brn (cl!n)} (P n) ∧ {} ∉ P n*
**and** *II*: *II ∈ UNpart1 cl dl P*
**shows** *BrnFT cl dl ∩ II = {}*

⟨*proof*⟩

**lemma** *BrnFT-notIn-UNpart1*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** *BrnFT cl dl* ∉ *UNpart1 cl dl P*
⟨*proof*⟩

**lemma** *UNpart1-UNpart01*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** *UNpart1 cl dl P* = *UNpart01 cl dl P* − {*BrnFT cl dl*}
⟨*proof*⟩

**lemma** *part-UNpart1*[*simp*]:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*)
**shows** *part* (*BrnNFT cl dl*) (*UNpart1 cl dl P*)
⟨*proof*⟩

**lemma** *part-UNpart01*:
**assumes** *cl*: *properL cl* **and** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** *part* {..< *brnL cl* (*length cl*)} (*UNpart01 cl dl P*)
⟨*proof*⟩

**definition** *UNlift01* **where**
*UNlift01 cl dl P F II* ≡
 *if II* = *BrnFT cl dl*
  *then BrnFT dl cl*
  *else shift dl* (*pickT cl P II*) (*F* (*pickT cl P II*) (*back cl* (*pickT cl P II*) *II*))

**lemma** *UNlift01-BrnFT*[*simp*]:
*UNlift01 cl dl P F* (*BrnFT cl dl*) = *BrnFT dl cl*
⟨*proof*⟩

**lemma** *UNlift01-shift*[*simp*]:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *n*: *n* ∈ *theNFTBoth cl dl* **and** *I*: *I* ∈ *P n*
**shows** *UNlift01 cl dl P F* (*shift cl n I*) = *shift dl n* (*F n I*)
⟨*proof*⟩

**lemma** *UNlift01-inj-on-UNpart1*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* ⟹ *part* {..< *brn* (*dl*!*n*)} (*F n* ‘ (*P n*)) ∧ {} ∉ *F n* ‘

(*P n*)
**and** *F*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *inj-on* (*F n*) (*P n*)
**shows** *inj-on* (*UNlift01 cl dl P F*) (*UNpart1 cl dl P*) (**is** *inj-on ?G ?Q*)
⟨*proof*⟩

**lemma** *inj-on-singl*:
**assumes** *inj-on f A* **and** *a0* ∉ *A* **and** $\bigwedge$ *a*. *a* ∈ *A* $\Longrightarrow$ *f a* ≠ *f a0*
**shows** *inj-on f* ({*a0*} *Un A*)
⟨*proof*⟩

**lemma** *UNlift01-inj-on*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*)) ∧ {} ∉ *F n* '
(*P n*)
**and** *F*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *inj-on* (*F n*) (*P n*)
**shows** *inj-on* (*UNlift01 cl dl P F*) (*UNpart01 cl dl P*)
⟨*proof*⟩

**lemma** *UNlift01-UNpart1*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** (*UNlift01 cl dl P F*) ' (*UNpart1 cl dl P*) = *UNpart1 dl cl* (%*n*. *F n* ' (*P*
*n*)) (**is** *?G* ' *?Q* = *?R*)
⟨*proof*⟩

**lemma** *UNlift01-UNpart01*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**shows** (*UNlift01 cl dl P F*) ' (*UNpart01 cl dl P*) = *UNpart01 dl cl* (%*n*. *F n* ' (*P*
*n*))
⟨*proof*⟩

**lemma** *emp-UNlift01-UNpart1*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ {} ∉ *F n* ' (*P n*)
**shows** {} ∉ (*UNlift01 cl dl P F*) ' (*UNpart1 cl dl P*) (**is** {} ∉ *?R*)
⟨*proof*⟩

**lemma** *emp-UNlift01-UNpart01*:
**assumes** *l*: *length cl* = *length dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ {} ∉ *F n* ' (*P n*)
**shows** {} ∉ (*UNlift01 cl dl P F*) ' (*UNpart01 cl dl P* − {*BrnFT cl dl*})
(**is** {} ∉ *?U* ' *?V*)
⟨*proof*⟩

**lemma** *part-UNlift01-UNpart1*:

**assumes** *l*: *length cl* = *length dl* **and** *dl*: *properL dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ *part* {..< *brn* (*dl*!*n*)} (*F n* ‘ (*P n*))
**shows** *part* (*BrnNFT dl cl*) ((*UNlift01 cl dl P F*) ‘ (*UNpart1 cl dl P*)) (**is** *part ?C ?R*)
⟨*proof*⟩

**lemma** *part-UNlift01-UNpart01*:
**assumes** *l*: *length cl* = *length dl* **and** *dl*: *properL dl*
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ *part* {..< *brn* (*dl*!*n*)} (*F n* ‘ (*P n*)) ∧ {} ∉ (*F n* ‘ (*P n*))
**shows** *part* {..< *brnL dl* (*length dl*)} ((*UNlift01 cl dl P F*) ‘ (*UNpart01 cl dl P*)) (**is** *part ?K ?R*)
⟨*proof*⟩

**lemma** *diff-frac-eq-1*:
**assumes** *b* ≠ (*0*::*real*)
**shows** *1* − *a* / *b* = (*b* − *a*) / *b*
⟨*proof*⟩

**lemma** *diff-frac-eq-2*:
**assumes** *b* ≠ (*1*::*real*)
**shows** *1* − (*a* − *b*) / (*1* − *b*) = (*1* − *a*) / (*1* − *b*)
(**is** *?L* = *?R*)
⟨*proof*⟩

**lemma** *triv-div-mult*:
**assumes** *vSF*: *vSF* ≠ (*1*::*real*)
**and** *L*: *L* = (*K* − *vSF*) / (*1* − *vSF*) **and** *Ln*: *L* ≠ *1*
**shows** (*VS* / (*1* − *vSF*) ∗ *V*) / (*1* − *L*) = (*VS* ∗ *V*) / (*1* − *K*)
(**is** *?A* = *?B*)
⟨*proof*⟩

**lemma** *ss-wt-ParT-UNlift01*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl* **and** *II*: *II* ∈ *UNpart01 cl dl P* − {*BrnFT cl dl*}
**and** *P*: $\bigwedge$ *n*. *n* < *length cl* $\Longrightarrow$ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*
**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ *part* {..< *brn* (*dl*!*n*)} (*F n* ‘ (*P n*))
**and** *sw*:
$\bigwedge$*n I*. ⟦*n* < *length cl*; *I* ∈ *P n*⟧ $\Longrightarrow$
  *sum* (*wt* (*cl* ! *n*) *s*) *I* =
  *sum* (*wt* (*dl* ! *n*) *t*) (*F n I*)
**and** *st*: *s* ≈ *t*
**and** *le1*: *sum* (*wt* (*ParT cl*) *s*) (*BrnFT cl dl*) < *1*
*sum* (*wt* (*ParT dl*) *t*) (*BrnFT dl cl*) < *1*
**shows**

*sum* (*wt* (*ParT cl*) *s*) *II* /
 (*1* − *sum* (*wt* (*ParT cl*) *s*) (*BrnFT cl dl*)) =
 *sum* (*wt* (*ParT dl*) *t*) (*UNlift01 cl dl P F II*) /
 (*1* − *sum* (*wt* (*ParT dl*) *t*) (*BrnFT dl cl*))
(**is** *sum ?vP II* / (*1* − *sum ?vP ?II-0*) =
 *sum ?wP ?JJ* / (*1* − *sum ?wP ?JJ-0*))
⟨*proof*⟩

**definition** *thetaZOParT* **where**
*thetaZOParT* ≡
 {(*ParT cl*, *ParT dl*) |
 *cl dl*.
 *properL cl* ∧ *properL dl* ∧ *SbisL cl dl*}

**lemma** *cont-eff-ParT-BrnFT-L*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl SbisL cl dl*
**and** *ii*: *ii* ∈ *BrnFT cl dl*
**and** *eff-cont*:
⋀*n I i j*. ⟦*n* < *length cl*; *I* ∈ *P n*; *i* ∈ *I*; *j* ∈ *F n I*⟧ ⟹
 *eff* (*cl*!*n*) *s i* ≈ *eff* (*dl*!*n*) *t j* ∧
 *cont* (*cl*!*n*) *s i* ≈*s cont* (*dl*!*n*) *t j*
**shows**
*s* ≈ *eff* (*ParT cl*) *s ii* ∧
 (*cont* (*ParT cl*) *s ii*, *ParT dl*) ∈ *thetaZOParT*
(**is** *?eff* ∧ *?cont*)
⟨*proof*⟩

**lemma** *cont-eff-ParT-BrnFT-R*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl SbisL cl dl*
**and** *jj*: *jj* ∈ *BrnFT dl cl*
**and** *eff-cont*:
⋀*n I i j*. ⟦*n* < *length cl*; *I* ∈ *P n*; *i* ∈ *I*; *j* ∈ *F n I*⟧ ⟹
 *eff* (*cl*!*n*) *s i* ≈ *eff* (*dl*!*n*) *t j* ∧ *cont* (*cl*!*n*) *s i* ≈*s cont* (*dl*!*n*) *t j*
**shows**
*t* ≈ *eff* (*ParT dl*) *t jj* ∧
 (*ParT cl*, *cont* (*ParT dl*) *t jj*) ∈ *thetaZOParT*
(**is** *?eff* ∧ *?cont*)
⟨*proof*⟩

**lemma** *cont-eff-ParT-UNlift01*:
**assumes** *l*: *length cl* = *length dl*
**and** *cldl*: *properL cl properL dl SbisL cl dl*
**and** *II*: *II* ∈ *UNpart01 cl dl P* − {*BrnFT cl dl*}
**and** *ii*: *ii* ∈ *II* **and** *jj*: *jj* ∈ *UNlift01 cl dl P F II*
**and** *P*: ⋀ *n*. *n* < *length cl* ⟹ *part* {..< *brn* (*cl*!*n*)} (*P n*) ∧ {} ∉ *P n*

**and** *FP*: $\bigwedge$ *n*. *n* < *length dl* $\Longrightarrow$ *part* {..< *brn* (*dl*!*n*)} (*F n* ' (*P n*))
**and** *eff-cont*:
$\bigwedge$*n I i j*. ⟦*n* < *length cl*; *I* ∈ *P n*; *i* ∈ *I*; *j* ∈ *F n I*⟧ $\Longrightarrow$
  *eff* (*cl*!*n*) *s i* ≈
  *eff* (*dl*!*n*) *t j* ∧
  *cont* (*cl*!*n*) *s i* ≈*s*
  *cont* (*dl*!*n*) *t j*
**and** *st*: *s* ≈ *t*
**shows**
*eff* (*ParT cl*) *s ii* ≈ *eff* (*ParT dl*) *t jj* ∧
 (*cont* (*ParT cl*) *s ii*, *cont* (*ParT dl*) *t jj*) ∈ *thetaZOParT*
(**is** *?eff* ∧ *?cont*)
⟨*proof*⟩

**lemma** *thetaZOParT-ZOretr*: *thetaZOParT* ⊆ *ZOretr* (*thetaZOParT*)
⟨*proof*⟩

**lemma** *thetaZOParT-ZObis*: *thetaZOParT* ⊆ *ZObis*
⟨*proof*⟩

**theorem** *ParT-ZObis*[*simp*]:
**assumes** *properL cl* **and** *properL dl* **and** *SbisL cl dl*
**shows** *ParT cl* ≈*01 ParT dl*
⟨*proof*⟩

**end**

**end**

# 5 Syntactic Criteria

**theory** *Syntactic-Criteria*
  **imports** *Compositionality*
**begin**

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

**lemma** *proper-intros*[*intro*]:
  *proper Done*
  *proper* (*Atm atm*)
  *proper c1* $\Longrightarrow$ *proper c2* $\Longrightarrow$ *proper* (*Seq c1 c2*)
  *proper c1* $\Longrightarrow$ *proper c2* $\Longrightarrow$ *proper* (*Ch ch c1 c2*)
  *proper c* $\Longrightarrow$ *proper* (*While tst c*)
  *properL cs* $\Longrightarrow$ *proper* (*Par cs*)
  *properL cs* $\Longrightarrow$ *proper* (*ParT cs*)

$(\bigwedge c.\ c \in set\ cs \Longrightarrow proper\ c) \Longrightarrow cs \neq [] \Longrightarrow properL\ cs$
⟨*proof*⟩

**lemma** *discr*:
$discr\ Done$
$presAtm\ atm \Longrightarrow discr\ (Atm\ atm)$
$discr\ c1 \Longrightarrow discr\ c2 \Longrightarrow discr\ (Seq\ c1\ c2)$
$discr\ c1 \Longrightarrow discr\ c2 \Longrightarrow discr\ (Ch\ ch\ c1\ c2)$
$discr\ c \Longrightarrow discr\ (While\ tst\ c)$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow discr\ c) \Longrightarrow discr\ (Par\ cs)$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow discr\ c) \Longrightarrow discr\ (ParT\ cs)$
⟨*proof*⟩

**lemma** *siso*:
$compatAtm\ atm \Longrightarrow siso\ (Atm\ atm)$
$siso\ c1 \Longrightarrow siso\ c2 \Longrightarrow siso\ (Seq\ c1\ c2)$
$compatCh\ ch \Longrightarrow siso\ c1 \Longrightarrow siso\ c2 \Longrightarrow siso\ (Ch\ ch\ c1\ c2)$
$compatTst\ tst \Longrightarrow siso\ c \Longrightarrow siso\ (While\ tst\ c)$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow siso\ c) \Longrightarrow siso\ (Par\ cs)$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow siso\ c) \Longrightarrow siso\ (ParT\ cs)$
⟨*proof*⟩

**lemma** *Sbis*:
$compatAtm\ atm \Longrightarrow Atm\ atm \approx s\ Atm\ atm$
$siso\ c1 \Longrightarrow c2 \approx s\ c2 \Longrightarrow Seq\ c1\ c2 \approx s\ Seq\ c1\ c2$
$proper\ c1 \Longrightarrow proper\ c2 \Longrightarrow c1 \approx s\ c1 \Longrightarrow discr\ c2 \Longrightarrow Seq\ c1\ c2 \approx s\ Seq\ c1\ c2$
$compatCh\ ch \Longrightarrow c1 \approx s\ c1 \Longrightarrow c2 \approx s\ c2 \Longrightarrow Ch\ ch\ c1\ c2 \approx s\ Ch\ ch\ c1\ c2$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow c \approx s\ c) \Longrightarrow Par\ cs \approx s\ Par\ cs$
⟨*proof*⟩

**lemma** *ZObis*:
$compatAtm\ atm \Longrightarrow Atm\ atm \approx 01\ Atm\ atm$
$siso\ c1 \Longrightarrow c2 \approx 01\ c2 \Longrightarrow Seq\ c1\ c2 \approx 01\ Seq\ c1\ c2$
$proper\ c1 \Longrightarrow proper\ c2 \Longrightarrow c1 \approx 01\ c1 \Longrightarrow discr\ c2 \Longrightarrow Seq\ c1\ c2 \approx 01\ Seq\ c1\ c2$
$compatCh\ ch \Longrightarrow c1 \approx 01\ c1 \Longrightarrow c2 \approx 01\ c2 \Longrightarrow Ch\ ch\ c1\ c2 \approx 01\ Ch\ ch\ c1\ c2$
$properL\ cs \Longrightarrow (\bigwedge c.\ c \in set\ cs \Longrightarrow c \approx s\ c) \Longrightarrow ParT\ cs \approx 01\ ParT\ cs$
⟨*proof*⟩

**lemma** *discr-imp-Sbis*: $proper\ c \Longrightarrow discr\ c \Longrightarrow c \approx s\ c$
⟨*proof*⟩

**lemma** *siso-imp-Sbis*: $siso\ c \Longrightarrow c \approx s\ c$
⟨*proof*⟩

**lemma** *Sbis-imp-ZObis*: $c \approx s\ c \Longrightarrow c \approx 01\ c$
⟨*proof*⟩

**fun** *SC-discr* **where**
*SC-discr Done* ⟷ *True*
| *SC-discr* (*Atm atm*) ⟷ *presAtm atm*
| *SC-discr* (*Seq c1 c2*) ⟷ *SC-discr c1* ∧ *SC-discr c2*
| *SC-discr* (*Ch ch c1 c2*) ⟷ *SC-discr c1* ∧ *SC-discr c2*
| *SC-discr* (*While tst c*) ⟷ *SC-discr c*
| *SC-discr* (*ParT cs*) ⟷ (∀ *c*∈*set cs*. *SC-discr c*)
| *SC-discr* (*Par cs*) ⟷ (∀ *c*∈*set cs*. *SC-discr c*)

**theorem** *SC-discr-discr*[*intro*]: *proper c* ⟹ *SC-discr c* ⟹ *discr c*
⟨*proof*⟩

**fun** *SC-siso* **where**
*SC-siso Done* ⟷ *True*
| *SC-siso* (*Atm atm*) ⟷ *compatAtm atm*
| *SC-siso* (*Seq c1 c2*) ⟷ *SC-siso c1* ∧ *SC-siso c2*
| *SC-siso* (*Ch ch c1 c2*) ⟷ *compatCh ch* ∧ *SC-siso c1* ∧ *SC-siso c2*
| *SC-siso* (*While tst c*) ⟷ *compatTst tst* ∧ *SC-siso c*
| *SC-siso* (*Par cs*) ⟷ (∀ *c*∈*set cs*. *SC-siso c*)
| *SC-siso* (*ParT cs*) ⟷ (∀ *c*∈*set cs*. *SC-siso c*)

**theorem** *SC-siso-siso*[*intro*]: *proper c* ⟹ *SC-siso c* ⟹ *siso c*
⟨*proof*⟩

**fun** *SC-Sbis* **where**
*SC-Sbis Done* ⟷ *True*
| *SC-Sbis* (*Atm atm*) ⟷ *compatAtm atm*
| *SC-Sbis* (*Seq c1 c2*) ⟷ (*SC-siso c1* ∧ *SC-Sbis c2*) ∨
(*SC-Sbis c1* ∧ *SC-discr c2*) ∨
*SC-discr* (*Seq c1 c2*) ∨ *SC-siso* (*Seq c1 c2*)
| *SC-Sbis* (*Ch ch c1 c2*) ⟷ (*if compatCh ch*
*then SC-Sbis c1* ∧ *SC-Sbis c2*
*else* (*SC-discr* (*Ch ch c1 c2*) ∨ *SC-siso* (*Ch ch c1 c2*)))
| *SC-Sbis* (*While tst c*) ⟷ *SC-discr* (*While tst c*) ∨ *SC-siso* (*While tst c*)
| *SC-Sbis* (*Par cs*) ⟷ (∀ *c*∈*set cs*. *SC-Sbis c*)
| *SC-Sbis* (*ParT cs*) ⟷ *SC-siso* (*ParT cs*) ∨ *SC-discr* (*ParT cs*)

**theorem** *SC-siso-SCbis*[*intro*]: *SC-siso c* ⟹ *SC-Sbis c*
⟨*proof*⟩

**theorem** *SC-discr-SCbis*[*intro*]: *SC-discr c* ⟹ *SC-Sbis c*
⟨*proof*⟩

**declare** *SC-siso.simps*[*simp del*]

**declare** *SC-discr.simps*[*simp del*]

**theorem** *SC-Sbis-Sbis*[*intro*]: *proper c* $\Longrightarrow$ *SC-Sbis c* $\Longrightarrow$ *c* $\approx$*s c*
⟨*proof*⟩

**fun** *SC-ZObis* **where**
*SC-ZObis Done* $\longleftrightarrow$ *True*
| *SC-ZObis* (*Atm atm*) $\longleftrightarrow$ *compatAtm atm*
| *SC-ZObis* (*Seq c1 c2*) $\longleftrightarrow$ (*SC-siso c1* $\wedge$ *SC-ZObis c2*) $\vee$
(*SC-ZObis c1* $\wedge$ *SC-discr c2*) $\vee$
*SC-Sbis* (*Seq c1 c2*)
| *SC-ZObis* (*Ch ch c1 c2*) $\longleftrightarrow$ (*if compatCh ch*
*then SC-ZObis c1* $\wedge$ *SC-ZObis c2*
*else SC-Sbis* (*Ch ch c1 c2*))
| *SC-ZObis* (*While tst c*) $\longleftrightarrow$ *SC-Sbis* (*While tst c*)
| *SC-ZObis* (*Par cs*) $\longleftrightarrow$ *SC-Sbis* (*Par cs*)
| *SC-ZObis* (*ParT cs*) $\longleftrightarrow$ ($\forall$ *c*$\in$*set cs. SC-Sbis c*)

**theorem** *SC-Sbis-SC-ZObis*[*intro*]: *SC-Sbis c* $\Longrightarrow$ *SC-ZObis c*
⟨*proof*⟩

**declare** *SC-Sbis.simps*[*simp del*]

**theorem** *SC-ZObis-ZObis*: *proper c* $\Longrightarrow$ *SC-ZObis c* $\Longrightarrow$ *c* $\approx$*01 c*
⟨*proof*⟩

**end**

**end**

# 6 Concrete setting

**theory** *Concrete*
**imports** *Syntactic-Criteria*
**begin**

**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* ≡ *Hi*
  **definition** *bot-level*: *bot* ≡ *Lo*
  **definition** *inf-level*: *inf l1 l2* ≡ *if Lo* ∈ {*l1*,*l2*} *then Lo else Hi*
  **definition** *sup-level*: *sup l1 l2* ≡ *if Hi* ∈ {*l1*,*l2*} *then Hi else Lo*
  **definition** *less-eq-level*: *less-eq l1 l2* ≡ (*l1* = *Lo* ∨ *l2* = *Hi*)
  **definition** *less-level*: *less l1 l2* ≡ *l1* = *Lo* ∧ *l2* = *Hi*
  **definition** *Inf-level*: *Inf L* ≡ *if Lo* ∈ *L then Lo else Hi*
  **definition** *Sup-level*: *Sup L* ≡ *if Hi* ∈ *L then Hi else Lo*
**instance**
  ⟨*proof*⟩
**end**

**lemma** *sup-eq-Lo*[*simp*]: *sup a b* = *Lo* ⟷ *a* = *Lo* ∧ *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** *choice* = *real* + *test*
**type-synonym** *state* = *var* ⇒ *nat*

**syntax**
 *-assign* :: *'a* ⇒ *'a* ⇒ *'a* (‹- ::= -› [*1000*, *61*] *61*)

**syntax-consts**
  *-assign* == *Assign*

**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* = (¬ *tval e s*)

**fun** *aval* **where**
*aval* (*Assign x e*) *s* = (*s* (*x* := *eval e s*))

**fun** *cval* **where**
*cval* (*Inl p*) *s* = *min 1* (*max 0 p*)
|*cval* (*Inr tst*) *s* = (*if tval tst s then 1 else 0*)

**definition** *indis* :: (*state* ∗ *state*) *set***where**
*indis* ≡ {(*s*,*t*). *ALL x. sec x* = *Lo* ⟶ *s x* = *t x*}

**interpretation** *Example-PL*: *PL-Indis aval tval cval indis*
⟨*proof*⟩

**fun** *exprSec* **where**
*exprSec* (*Ct n*) = *Lo*
|*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* = *Lo*
|*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* ⟹ (*s*, *t*) ∈ *indis* ⟹ *eval expr s* = *eval expr t*
⟨*proof*⟩

**lemma** *compatAtmSyntactic*[*simp*]: *exprSec expr* = *Lo* ∨ *sec v* = *Hi* ⟹ *Example-PL.compatAtm* (*Assign v expr*)
⟨*proof*⟩

**lemma** *presAtmSyntactic*[*simp*]: *sec v* = *Hi* ⟹ *Example-PL.presAtm* (*Assign v expr*)
⟨*proof*⟩

**lemma** *compatTstSyntactic*[*simp*]: *tstSec tst* = *Lo* ⟹ *Example-PL.compatTst tst*
⟨*proof*⟩

**lemma** *compatPrchSyntactic*[*simp*]: *Example-PL.compatCh* (*Inl p*)
⟨*proof*⟩

**lemma** *compatIfchSyntactic*[*simp*]: *Example-PL.compatCh* (*Inr tst*) ⟷ *Example-PL.compatTst tst*
⟨*proof*⟩

**abbreviation** *Ch-half* (‹$Ch_{½}$›) **where** $Ch_{½}$ ≡ *Ch* (*Inl* (*1/2*))
**abbreviation** *If* **where** *If tst* ≡ *Ch* (*Inr tst*)

**abbreviation** *siso c* ≡ *Example-PL.siso c*
**abbreviation** *discr c* ≡ *Example-PL.discr c*
**abbreviation** *Sbis-abbrev* (**infix** ‹≈s› *55*) **where** *c1* ≈*s c2* ≡ (*c1*,*c2*) ∈ *Example-PL.Sbis*
**abbreviation** *ZObis-abbrev* (**infix** ‹≈01› *55*) **where** *c1* ≈*01 c2* ≡ (*c1*,*c2*) ∈ *Example-PL.ZObis*

**abbreviation** *SC-siso c* ≡ *Example-PL.SC-siso c*
**abbreviation** *SC-discr c* ≡ *Example-PL.SC-discr c*
**abbreviation** *SC-Sbis c* ≡ *Example-PL.SC-Sbis c*
**abbreviation** *SC-ZObis c* ≡ *Example-PL.SC-ZObis c*

**lemma** *SC-discr* (*h* ::= *Ct 0*)
⟨*proof*⟩

## 6.1 The secure programs from the paper's Example 3

**definition** [*simp*]: *d0* =
*h′* ::= *Ct 0* ;;
*While* (*Gt* (*Var h*) (*Ct 0*))
($Ch_{½}$ (*h* ::= *Ct 0*)
(*h′* ::= *Plus* (*Var h′*) (*Ct 1*)))

**definition** [*simp*]: *d1* =
*While* (*Gt* (*Var h*) (*Ct 0*))
($Ch_{½}$ (*h* ::= *Minus* (*Var h*) (*Ct 1*))
(*h* ::= *Plus* (*Var h*) (*Ct 1*)))

**definition** [*simp*]: *d2* =
*If* (*Eq* (*Var l*) (*Ct 0*))
(*l′* ::= *Ct 1*)
*d0*

**definition** [*simp*]: *d3* =
*h* ::= *Ct 5* ;;
*ParT* [*d0*, (*l* ::= *Ct 1*)]

**theorem** *SC-discr d0*
*SC-discr d1*

*SC-Sbis d2*
*SC-ZObis d2*
⟨*proof*⟩

**theorem** *discr d0*
*discr d1*
*d2* $\approx$*s d2*
*d3* $\approx$*01 d3*
⟨*proof*⟩

**end**

# References


[1] A. Popescu, J. Hölzl, and T. Nipkow. Formalizing probabilistic noninterference. In G. Gonthier and M. Norrish, editors, *Certified Programs and Proofs (CPP 2013)*, volume 8307 of *LNCS*, pages 259–275. Springer, 2013.

[2] A. Popescu, J. Hölzl, and T. Nipkow. Noninterfering schedulers. In R. Heckel and S. Milius, editors, *Algebra and Coalgebra in Computer Science (CALCO 2013)*, volume 8089 of *LNCS*, pages 236–252. Springer, 2013.