

The Calculus of Communicating Systems

Jesper Bengtson

February 6, 2026

Abstract

We formalise a large portion of CCS as described in Milner’s book ‘Communication and Concurrency’ using the nominal datatype package in Isabelle. Our results include many of the standard theorems of bisimulation equivalence and congruence, for both weak and strong versions. One main goal of this formalisation is to keep the machine-checked proofs as close to their pen-and-paper counterpart as possible.

Contents

1 Overview	1
2 Formalisation	2

1 Overview

These theories formalise the following results from Milner’s book Communication and Concurrency.

- strong bisimilarity is a congruence
- strong bisimilarity respects the laws of structural congruence
- weak bisimilarity is preserved by all operators except sum
- weak congruence is a congruence
- all strongly bisimilar agents are also weakly congruent which in turn are weakly bisimilar. As a corollary, weak bisimilarity and weak congruence respect the laws of structural congruence.

The file naming convention is hopefully self explanatory, where the prefixes *Strong* and *Weak* denote that the file covers theories required to formalise properties of strong and weak bisimilarity respectively; if the file name contains *Sim* the theories cover simulation, file names containing *Bisim*

cover bisimulation, and file names containing *Cong* cover weak congruence; files with the suffix *Pres* deal with theories that reason about preservation properties of operators such as a certain simulation or bisimulation being preserved by a certain operator; files with the suffix *SC* reason about structural congruence.

For a complete exposition of all theories, please consult Bengtson's Ph. D. thesis [1].

2 Formalisation

```

theory Agent
  imports HOL-Nominal.Nominal
begin

atom-decl name

nominal-datatype act = actAction name      (⟨!-⟩ 100)
  | actCoAction name    (⟨⟨-⟩ 100)
  | actTau               (⟨τ⟩ 100)

nominal-datatype ccs = CCSNil              (⟨0⟩ 115)
  | Action act ccs    (⟨-.-⟩ [120, 110] 110)
  | Sum ccs ccs       (infixl ⟨⊕⟩ 90)
  | Par ccs ccs       (infixl ⟨||⟩ 85)
  | Res «name» ccs   (⟨!ν-⟩ [105, 100] 100)
  | Bang ccs         (⟨!-⟩ [95])

nominal-primrec coAction :: act ⇒ act
where
  coAction (|a|) = (⟨a⟩)
  | coAction (⟨a⟩) = (|a|)
  | coAction (τ) = τ
  ⟨proof⟩

lemma coActionEqvt[eqvt]:
  fixes p :: name prm
  and a :: act

  shows (p · coAction a) = coAction(p · a)
  ⟨proof⟩

lemma coActionSimps[simp]:
  fixes a :: act

  shows coAction(coAction a) = a
  and (coAction a = τ) = (a = τ)
  ⟨proof⟩

```

lemma *coActSimp*[simp]: **shows** $\text{coAction } \alpha \neq \tau = (\alpha \neq \tau)$ **and** $(\text{coAction } \alpha = \tau) = (\alpha = \tau)$
 ⟨proof⟩

lemma *coActFresh*[simp]:

fixes $x :: \text{name}$
and $a :: \text{act}$

shows $x \# \text{coAction } a = x \# a$
 ⟨proof⟩

lemma *alphaRes*:

fixes $y :: \text{name}$
and $P :: \text{ccs}$
and $x :: \text{name}$

assumes $y \# P$

shows $(\nu x)P = (\nu y)((x, y) \cdot P)$
 ⟨proof⟩

inductive semantics :: $\text{ccs} \Rightarrow \text{act} \Rightarrow \text{ccs} \Rightarrow \text{bool}$ ($\langle \cdot \rangle \mapsto \cdot \prec \rightarrow$ [80, 80, 80] 80)

where

Action: $\alpha.(P) \mapsto \alpha \prec P$
 | *Sum1*: $P \mapsto \alpha \prec P' \Longrightarrow P \oplus Q \mapsto \alpha \prec P'$
 | *Sum2*: $Q \mapsto \alpha \prec Q' \Longrightarrow P \oplus Q \mapsto \alpha \prec Q'$
 | *Par1*: $P \mapsto \alpha \prec P' \Longrightarrow P \parallel Q \mapsto \alpha \prec P' \parallel Q$
 | *Par2*: $Q \mapsto \alpha \prec Q' \Longrightarrow P \parallel Q \mapsto \alpha \prec P \parallel Q'$
 | *Comm*: $\llbracket P \mapsto a \prec P'; Q \mapsto (\text{coAction } a) \prec Q'; a \neq \tau \rrbracket \Longrightarrow P \parallel Q \mapsto \tau \prec P' \parallel Q'$
 | *Res*: $\llbracket P \mapsto \alpha \prec P'; x \# \alpha \rrbracket \Longrightarrow (\nu x)P \mapsto \alpha \prec (\nu x)P'$
 | *Bang*: $P \parallel !P \mapsto \alpha \prec P' \Longrightarrow !P \mapsto \alpha \prec P'$

equivariance semantics

nominal-inductive semantics

⟨proof⟩

lemma *semanticsInduct*:

$\llbracket R \mapsto \beta \prec R'; \bigwedge \alpha P \mathcal{C}. \text{Prop } \mathcal{C} (\alpha.(P)) \alpha P;$
 $\bigwedge P \alpha P' Q \mathcal{C}. \llbracket P \mapsto \alpha \prec P'; \bigwedge \mathcal{C}. \text{Prop } \mathcal{C} P \alpha P' \rrbracket \Longrightarrow \text{Prop } \mathcal{C} (\text{ccs.Sum } P Q) \alpha P';$
 $\bigwedge Q \alpha Q' P \mathcal{C}. \llbracket Q \mapsto \alpha \prec Q'; \bigwedge \mathcal{C}. \text{Prop } \mathcal{C} Q \alpha Q' \rrbracket \Longrightarrow \text{Prop } \mathcal{C} (\text{ccs.Sum } P Q) \alpha Q';$
 $\bigwedge P \alpha P' Q \mathcal{C}. \llbracket P \mapsto \alpha \prec P'; \bigwedge \mathcal{C}. \text{Prop } \mathcal{C} P \alpha P' \rrbracket \Longrightarrow \text{Prop } \mathcal{C} (P \parallel Q) \alpha (P' \parallel Q);$
 $\bigwedge Q \alpha Q' P \mathcal{C}. \llbracket Q \mapsto \alpha \prec Q'; \bigwedge \mathcal{C}. \text{Prop } \mathcal{C} Q \alpha Q' \rrbracket \Longrightarrow \text{Prop } \mathcal{C} (P \parallel Q) \alpha (P \parallel Q');$

$\wedge P a P' Q Q' \mathcal{C}.$
 $\llbracket P \mapsto a \prec P'; \wedge \mathcal{C}. Prop \mathcal{C} P a P'; Q \mapsto (coAction a) \prec Q';$
 $\wedge \mathcal{C}. Prop \mathcal{C} Q (coAction a) Q'; a \neq \tau \rrbracket$
 $\implies Prop \mathcal{C} (P \parallel Q) (\tau) (P' \parallel Q');$
 $\wedge P \alpha P' x \mathcal{C}.$
 $\llbracket x \# \mathcal{C}; P \mapsto \alpha \prec P'; \wedge \mathcal{C}. Prop \mathcal{C} P \alpha P'; x \# \alpha \rrbracket \implies Prop \mathcal{C} (\nu x) P \alpha$
 $(\nu x) P';$
 $\wedge P \alpha P' \mathcal{C}. \llbracket P \parallel !P \mapsto \alpha \prec P'; \wedge \mathcal{C}. Prop \mathcal{C} (P \parallel !P) \alpha P' \rrbracket \implies Prop \mathcal{C} !P \alpha P'$
 $\implies Prop (\mathcal{C}::'a::fs-name) R \beta R'$
 $\langle proof \rangle$

lemma *NilTrans[dest]*:
shows $0 \mapsto \alpha \prec P' \implies False$
and $(\langle b \rangle).P \mapsto \langle c \rangle \prec P' \implies False$
and $(\langle b \rangle).P \mapsto \tau \prec P' \implies False$
and $(\langle b \rangle).P \mapsto \langle c \rangle \prec P' \implies False$
and $(\langle b \rangle).P \mapsto \tau \prec P' \implies False$
 $\langle proof \rangle$

lemma *freshDerivative*:
fixes $P :: ccs$
and $a :: act$
and $P' :: ccs$
and $x :: name$

assumes $P \mapsto \alpha \prec P'$
and $x \# P$

shows $x \# \alpha$ **and** $x \# P'$
 $\langle proof \rangle$

lemma *actCases[consumes 1, case-names cAct]*:
fixes $\alpha :: act$
and $P :: ccs$
and $\beta :: act$
and $P' :: ccs$

assumes $\alpha.(P) \mapsto \beta \prec P'$
and $Prop \alpha P$

shows $Prop \beta P'$
 $\langle proof \rangle$

lemma *sumCases[consumes 1, case-names cSum1 cSum2]*:
fixes $P :: ccs$
and $Q :: ccs$
and $\alpha :: act$
and $R :: ccs$

assumes $P \oplus Q \mapsto \alpha \prec R$
and $\bigwedge P'. P \mapsto \alpha \prec P' \implies Prop P'$
and $\bigwedge Q'. Q \mapsto \alpha \prec Q' \implies Prop Q'$

shows $Prop R$
 $\langle proof \rangle$

lemma $parCases[consumes\ 1, case-names\ cPar1\ cPar2\ cComm]$:
fixes $P :: ccs$
and $Q :: ccs$
and $a :: act$
and $R :: ccs$

assumes $P \parallel Q \mapsto \alpha \prec R$
and $\bigwedge P'. P \mapsto \alpha \prec P' \implies Prop\ \alpha\ (P' \parallel Q)$
and $\bigwedge Q'. Q \mapsto \alpha \prec Q' \implies Prop\ \alpha\ (P \parallel Q')$
and $\bigwedge P' Q' a. \llbracket P \mapsto a \prec P'; Q \mapsto (coAction\ a) \prec Q'; a \neq \tau; \alpha = \tau \rrbracket \implies Prop\ (\tau)\ (P' \parallel Q')$

shows $Prop\ \alpha\ R$
 $\langle proof \rangle$

lemma $resCases[consumes\ 1, case-names\ cRes]$:
fixes $x :: name$
and $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$

assumes $(\nu x)P \mapsto \alpha \prec P'$
and $\bigwedge P'. \llbracket P \mapsto \alpha \prec P'; x \# \alpha \rrbracket \implies Prop\ ((\nu x)P')$

shows $Prop\ P'$
 $\langle proof \rangle$

inductive $bangPred :: ccs \Rightarrow ccs \Rightarrow bool$
where
 $aux1: bangPred\ P\ (!P)$
 $| aux2: bangPred\ P\ (P \parallel !P)$

lemma $bangInduct[consumes\ 1, case-names\ cPar1\ cPar2\ cComm\ cBang]$:
fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$
and $C :: 'a::fs-name$

assumes $!P \mapsto \alpha \prec P'$
and $rPar1: \bigwedge \alpha P' C. \llbracket P \mapsto \alpha \prec P' \rrbracket \implies Prop\ C\ (P \parallel !P)\ \alpha\ (P' \parallel !P)$
and $rPar2: \bigwedge \alpha P' C. \llbracket !P \mapsto \alpha \prec P'; \bigwedge C. Prop\ C\ (!P)\ \alpha\ P' \rrbracket \implies Prop\ C\ (P$

$\parallel !P) \alpha (P \parallel P')$
and $rComm: \bigwedge a P' P'' C. \llbracket P \mapsto a \prec P'; !P \mapsto (coAction\ a) \prec P''; \bigwedge C. Prop\ C\ (!P)\ (coAction\ a)\ P''; a \neq \tau \rrbracket \implies Prop\ C\ (P \parallel !P)\ (\tau)\ (P' \parallel P'')$
and $rBang: \bigwedge \alpha P' C. \llbracket P \parallel !P \mapsto \alpha \prec P'; \bigwedge C. Prop\ C\ (P \parallel !P)\ \alpha\ P' \rrbracket \implies Prop\ C\ (!P)\ \alpha\ P'$

shows $Prop\ C\ (!P)\ \alpha\ P'$
 $\langle proof \rangle$

inductive-set $bangRel :: (ccs \times ccs)\ set \implies (ccs \times ccs)\ set$
for $Rel :: (ccs \times ccs)\ set$

where

$BRBang: (P, Q) \in Rel \implies (!P, !Q) \in bangRel\ Rel$
 $| BRPar: (R, T) \in Rel \implies (P, Q) \in (bangRel\ Rel) \implies (R \parallel P, T \parallel Q) \in (bangRel\ Rel)$

lemma $BRBangCases[consumes\ 1, case-names\ BRBang]:$

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs)\ set$
and $F :: ccs \implies bool$

assumes $(P, !Q) \in bangRel\ Rel$
and $\bigwedge P. (P, Q) \in Rel \implies F\ (!P)$

shows $F\ P$
 $\langle proof \rangle$

lemma $BRParCases[consumes\ 1, case-names\ BRPar]:$

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs)\ set$
and $F :: ccs \implies bool$

assumes $(P, Q \parallel !Q) \in bangRel\ Rel$
and $\bigwedge P R. \llbracket (P, Q) \in Rel; (R, !Q) \in bangRel\ Rel \rrbracket \implies F\ (P \parallel R)$

shows $F\ P$
 $\langle proof \rangle$

lemma $bangRelSubset:$

fixes $Rel :: (ccs \times ccs)\ set$
and $Rel' :: (ccs \times ccs)\ set$

assumes $(P, Q) \in bangRel\ Rel$
and $\bigwedge P Q. (P, Q) \in Rel \implies (P, Q) \in Rel'$

shows $(P, Q) \in bangRel\ Rel'$

<proof>

end

theory *Tau-Chain*
imports *Agent*
begin

definition *tauChain* :: *ccs* \Rightarrow *ccs* \Rightarrow *bool* (λ $- \Rightarrow_{\tau} - \rightarrow [80, 80] 80$)
where $P \Rightarrow_{\tau} P' \equiv (P, P') \in \{(P, P') \mid P P'. P \mapsto_{\tau} \prec P'\}^{\widehat{*}}$

lemma *tauChainInduct*[*consumes 1, case-names Base Step*]:

assumes $P \Rightarrow_{\tau} P'$

and *Prop* P

and $\bigwedge P' P''. \llbracket P \Rightarrow_{\tau} P'; P' \mapsto_{\tau} \prec P''; \text{Prop } P' \rrbracket \Rightarrow \text{Prop } P''$

shows *Prop* P'

<proof>

lemma *tauChainRefl*[*simp*]:

fixes $P :: \text{ccs}$

shows $P \Rightarrow_{\tau} P$

<proof>

lemma *tauChainCons*[*dest*]:

fixes $P :: \text{ccs}$

and $P' :: \text{ccs}$

and $P'' :: \text{ccs}$

assumes $P \Rightarrow_{\tau} P'$

and $P' \mapsto_{\tau} \prec P''$

shows $P \Rightarrow_{\tau} P''$

<proof>

lemma *tauChainCons2*[*dest*]:

fixes $P :: \text{ccs}$

and $P' :: \text{ccs}$

and $P'' :: \text{ccs}$

assumes $P' \mapsto_{\tau} \prec P''$

and $P \Rightarrow_{\tau} P'$

shows $P \Rightarrow_{\tau} P''$

<proof>

lemma *tauChainAppend*[*dest*]:

```

fixes  $P :: ccs$ 
and  $P' :: ccs$ 
and  $P'' :: ccs$ 

assumes  $P \Longrightarrow_{\tau} P'$ 
and  $P' \Longrightarrow_{\tau} P''$ 

shows  $P \Longrightarrow_{\tau} P''$ 
<proof>

lemma tauChainSum1:
fixes  $P :: ccs$ 
and  $P' :: ccs$ 
and  $Q :: ccs$ 

assumes  $P \Longrightarrow_{\tau} P'$ 
and  $P \neq P'$ 

shows  $P \oplus Q \Longrightarrow_{\tau} P'$ 
<proof>

lemma tauChainSum2:
fixes  $P :: ccs$ 
and  $P' :: ccs$ 
and  $Q :: ccs$ 

assumes  $Q \Longrightarrow_{\tau} Q'$ 
and  $Q \neq Q'$ 

shows  $P \oplus Q \Longrightarrow_{\tau} Q'$ 
<proof>

lemma tauChainPar1:
fixes  $P :: ccs$ 
and  $P' :: ccs$ 
and  $Q :: ccs$ 

assumes  $P \Longrightarrow_{\tau} P'$ 

shows  $P \parallel Q \Longrightarrow_{\tau} P' \parallel Q$ 
<proof>

lemma tauChainPar2:
fixes  $Q :: ccs$ 
and  $Q' :: ccs$ 
and  $P :: ccs$ 

assumes  $Q \Longrightarrow_{\tau} Q'$ 

```

shows $P \parallel Q \Longrightarrow_{\tau} P \parallel Q'$
 ⟨*proof*⟩

lemma *tauChainRes*:

fixes $P :: ccs$
and $P' :: ccs$
and $x :: name$

assumes $P \Longrightarrow_{\tau} P'$

shows $(\nu x)P \Longrightarrow_{\tau} (\nu x)P'$
 ⟨*proof*⟩

lemma *tauChainRepl*:

fixes $P :: ccs$

assumes $P \parallel !P \Longrightarrow_{\tau} P'$
and $P' \neq P \parallel !P$

shows $!P \Longrightarrow_{\tau} P'$
 ⟨*proof*⟩

end

theory *Weak-Cong-Semantics*

imports *Tau-Chain*

begin

definition *weakCongTrans* :: $ccs \Rightarrow act \Rightarrow ccs \Rightarrow bool$ ($\langle - \Longrightarrow - \prec - \rangle [80, 80, 80]$
 80)

where $P \Longrightarrow_{\alpha} \prec P' \equiv \exists P'' P'''. P \Longrightarrow_{\tau} P'' \wedge P'' \mapsto_{\alpha} \prec P''' \wedge P''' \Longrightarrow_{\tau} P'$

lemma *weakCongTransE*:

fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$

assumes $P \Longrightarrow_{\alpha} \prec P'$

obtains $P'' P'''$ **where** $P \Longrightarrow_{\tau} P''$ **and** $P'' \mapsto_{\alpha} \prec P'''$ **and** $P''' \Longrightarrow_{\tau} P'$
 ⟨*proof*⟩

lemma *weakCongTransI*:

fixes $P :: ccs$
and $P'' :: ccs$
and $\alpha :: act$
and $P''' :: ccs$
and $P' :: ccs$

assumes $P \Longrightarrow_{\tau} P''$
and $P'' \mapsto_{\alpha} \prec P'''$
and $P''' \Longrightarrow_{\tau} P'$

shows $P \Longrightarrow_{\alpha} \prec P'$
<proof>

lemma *transitionWeakCongTransition:*

fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$

assumes $P \mapsto_{\alpha} \prec P'$

shows $P \Longrightarrow_{\alpha} \prec P'$
<proof>

lemma *weakCongAction:*

fixes $a :: name$
and $P :: ccs$

shows $\alpha.(P) \Longrightarrow_{\alpha} \prec P$
<proof>

lemma *weakCongSum1:*

fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$
and $Q :: ccs$

assumes $P \Longrightarrow_{\alpha} \prec P'$

shows $P \oplus Q \Longrightarrow_{\alpha} \prec P'$
<proof>

lemma *weakCongSum2:*

fixes $Q :: ccs$
and $\alpha :: act$
and $Q' :: ccs$
and $P :: ccs$

assumes $Q \Longrightarrow_{\alpha} \prec Q'$

shows $P \oplus Q \Longrightarrow_{\alpha} \prec Q'$
<proof>

lemma *weakCongPar1:*

fixes $P :: ccs$
and $\alpha :: act$

```

and P' :: ccs
and Q  :: ccs

assumes P  $\Longrightarrow$   $\alpha$  < P'

shows P || Q  $\Longrightarrow$   $\alpha$  < P' || Q
<proof>

lemma weakCongPar2:
fixes Q  :: ccs
and    $\alpha$  :: act
and   Q'  :: ccs
and   P  :: ccs

assumes Q  $\Longrightarrow$   $\alpha$  < Q'

shows P || Q  $\Longrightarrow$   $\alpha$  < P || Q'
<proof>

lemma weakCongSync:
fixes P  :: ccs
and    $\alpha$  :: act
and   P'  :: ccs
and   Q  :: ccs

assumes P  $\Longrightarrow$   $\alpha$  < P'
and   Q  $\Longrightarrow$  (coAction  $\alpha$ ) < Q'
and    $\alpha \neq \tau$ 

shows P || Q  $\Longrightarrow$   $\tau$  < P' || Q'
<proof>

lemma weakCongRes:
fixes P  :: ccs
and    $\alpha$  :: act
and   P'  :: ccs
and   x  :: name

assumes P  $\Longrightarrow$   $\alpha$  < P'
and   x #  $\alpha$ 

shows ( $\nu x$ )P  $\Longrightarrow$   $\alpha$  < ( $\nu x$ )P'
<proof>

lemma weakCongRepl:
fixes P  :: ccs
and    $\alpha$  :: act
and   P'  :: ccs

```

```

assumes  $P \parallel !P \Longrightarrow \alpha \prec P'$ 

shows  $!P \Longrightarrow \alpha \prec P'$ 
<proof>

end

theory Weak-Semantics
  imports Weak-Cong-Semantics
begin

definition weakTrans ::  $ccs \Rightarrow act \Rightarrow ccs \Rightarrow bool$  ( $\prec \Longrightarrow \hat{\phantom{x}} \prec \rightarrow$  [80, 80, 80] 80)
  where  $P \Longrightarrow \hat{\alpha} \prec P' \equiv P \Longrightarrow \alpha \prec P' \vee (\alpha = \tau \wedge P = P')$ 

lemma weakEmptyTrans[simp]:
  fixes  $P :: ccs$ 

  shows  $P \Longrightarrow \hat{\tau} \prec P$ 
<proof>

lemma weakTransCases[consumes 1, case-names Base Step]:
  fixes  $P :: ccs$ 
  and  $\alpha :: act$ 
  and  $P' :: ccs$ 

  assumes  $P \Longrightarrow \hat{\alpha} \prec P'$ 
  and  $\llbracket \alpha = \tau; P = P' \rrbracket \Longrightarrow Prop (\tau) P$ 
  and  $P \Longrightarrow \alpha \prec P' \Longrightarrow Prop \alpha P'$ 

  shows  $Prop \alpha P'$ 
<proof>

lemma weakCongTransitionWeakTransition:
  fixes  $P :: ccs$ 
  and  $\alpha :: act$ 
  and  $P' :: ccs$ 

  assumes  $P \Longrightarrow \alpha \prec P'$ 

  shows  $P \Longrightarrow \hat{\alpha} \prec P'$ 
<proof>

lemma transitionWeakTransition:
  fixes  $P :: ccs$ 
  and  $\alpha :: act$ 
  and  $P' :: ccs$ 

  assumes  $P \vdash \alpha \prec P'$ 

```

shows $P \Longrightarrow \hat{\alpha} \prec P'$
<proof>

lemma *weakAction*:

fixes $a :: name$
and $P :: ccs$

shows $\alpha.(P) \Longrightarrow \hat{\alpha} \prec P$
<proof>

lemma *weakSum1*:

fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$
and $Q :: ccs$

assumes $P \Longrightarrow \hat{\alpha} \prec P'$
and $P \neq P'$

shows $P \oplus Q \Longrightarrow \hat{\alpha} \prec P'$
<proof>

lemma *weakSum2*:

fixes $Q :: ccs$
and $\alpha :: act$
and $Q' :: ccs$
and $P :: ccs$

assumes $Q \Longrightarrow \hat{\alpha} \prec Q'$
and $Q \neq Q'$

shows $P \oplus Q \Longrightarrow \hat{\alpha} \prec Q'$
<proof>

lemma *weakPar1*:

fixes $P :: ccs$
and $\alpha :: act$
and $P' :: ccs$
and $Q :: ccs$

assumes $P \Longrightarrow \hat{\alpha} \prec P'$

shows $P \parallel Q \Longrightarrow \hat{\alpha} \prec P' \parallel Q$
<proof>

lemma *weakPar2*:

fixes $Q :: ccs$
and $\alpha :: act$
and $Q' :: ccs$

```

and P :: ccs

assumes Q  $\Longrightarrow$   $\hat{\alpha} \prec Q'$ 

shows P || Q  $\Longrightarrow$   $\hat{\alpha} \prec P || Q'$ 
<proof>

lemma weakSync:
fixes P :: ccs
and  $\alpha$  :: act
and P' :: ccs
and Q :: ccs

assumes P  $\Longrightarrow$   $\hat{\alpha} \prec P'$ 
and Q  $\Longrightarrow$   $\hat{(coAction \alpha)} \prec Q'$ 
and  $\alpha \neq \tau$ 

shows P || Q  $\Longrightarrow$   $\hat{\tau} \prec P' || Q'$ 
<proof>

lemma weakRes:
fixes P :: ccs
and  $\alpha$  :: act
and P' :: ccs
and x :: name

assumes P  $\Longrightarrow$   $\hat{\alpha} \prec P'$ 
and x  $\# \alpha$ 

shows ( $\nu x$ )P  $\Longrightarrow$   $\hat{\alpha} \prec (\nu x)P'$ 
<proof>

lemma weakRepl:
fixes P :: ccs
and  $\alpha$  :: act
and P' :: ccs

assumes P || !P  $\Longrightarrow$   $\hat{\alpha} \prec P'$ 
and P'  $\neq P || !P$ 

shows !P  $\Longrightarrow$   $\alpha \prec P'$ 
<proof>

end

theory Strong-Sim
imports Agent
begin

```

definition *simulation* :: $ccs \Rightarrow (ccs \times ccs) \text{ set} \Rightarrow ccs \Rightarrow \text{bool}$ ($\langle \sim \rangle [-] \rightarrow [80, 80, 80] 80$)

where

$P \sim \langle Rel \rangle Q \equiv \forall a Q'. Q \mapsto a \prec Q' \longrightarrow (\exists P'. P \mapsto a \prec P' \wedge (P', Q') \in Rel)$

lemma *simI*[*case-names Sim*]:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $\bigwedge \alpha Q'. Q \mapsto \alpha \prec Q' \implies \exists P'. P \mapsto \alpha \prec P' \wedge (P', Q') \in Rel$

shows $P \sim \langle Rel \rangle Q$

<proof>

lemma *simE*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $\alpha :: \text{act}$
and $Q' :: ccs$

assumes $P \sim \langle Rel \rangle Q$
and $Q \mapsto \alpha \prec Q'$

obtains P' **where** $P \mapsto \alpha \prec P'$ **and** $(P', Q') \in Rel$

<proof>

lemma *reflexive*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $Id \subseteq Rel$

shows $P \sim \langle Rel \rangle P$

<proof>

lemma *transitive*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $Rel' :: (ccs \times ccs) \text{ set}$
and $R :: ccs$
and $Rel'' :: (ccs \times ccs) \text{ set}$

assumes $P \sim \langle Rel \rangle Q$
and $Q \sim \langle Rel' \rangle R$
and $Rel \circ Rel' \subseteq Rel''$

shows $P \rightsquigarrow[Rel'] R$
 ⟨proof⟩

end

theory *Weak-Sim*
imports *Weak-Semantics Strong-Sim*
begin

definition *weakSimulation* :: $ccs \Rightarrow (ccs \times ccs) \text{ set} \Rightarrow ccs \Rightarrow \text{bool}$ ($\langle - \rightsquigarrow^{\wedge} \langle - \rangle - \rangle$
 [80, 80, 80] 80)

where

$P \rightsquigarrow^{\wedge} \langle Rel \rangle Q \equiv \forall a Q'. Q \mapsto a \prec Q' \longrightarrow (\exists P'. P \Longrightarrow^{\wedge} a \prec P' \wedge (P', Q') \in Rel)$

lemma *weakSimI*[*case-names Sim*]:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $\bigwedge \alpha Q'. Q \mapsto \alpha \prec Q' \Longrightarrow \exists P'. P \Longrightarrow^{\wedge} \alpha \prec P' \wedge (P', Q') \in Rel$

shows $P \rightsquigarrow^{\wedge} \langle Rel \rangle Q$
 ⟨proof⟩

lemma *weakSimE*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $\alpha :: \text{act}$
and $Q' :: ccs$

assumes $P \rightsquigarrow^{\wedge} \langle Rel \rangle Q$
and $Q \mapsto \alpha \prec Q'$

obtains P' **where** $P \Longrightarrow^{\wedge} \alpha \prec P'$ **and** $(P', Q') \in Rel$
 ⟨proof⟩

lemma *simTauChain*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $Q' :: ccs$

assumes $Q \Longrightarrow_{\tau} Q'$
and $(P, Q) \in Rel$
and $Sim: \bigwedge R S. (R, S) \in Rel \Longrightarrow R \rightsquigarrow^{\wedge} \langle Rel \rangle S$

obtains P' **where** $P \Longrightarrow_{\tau} P'$ **and** $(P', Q') \in Rel$

<proof>

lemma *simE2*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $\alpha :: act$
and $Q' :: ccs$

assumes $(P, Q) \in Rel$
and $Q \Longrightarrow \hat{\alpha} \prec Q'$
and $Sim: \bigwedge R S. (R, S) \in Rel \Longrightarrow R \rightsquigarrow \hat{\langle Rel \rangle} S$

obtains P' **where** $P \Longrightarrow \hat{\alpha} \prec P'$ **and** $(P', Q') \in Rel$
<proof>

lemma *reflexive*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $Id \subseteq Rel$

shows $P \rightsquigarrow \hat{\langle Rel \rangle} P$
<proof>

lemma *transitive*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $Rel' :: (ccs \times ccs) \text{ set}$
and $R :: ccs$
and $Rel'' :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in Rel$
and $Q \rightsquigarrow \hat{\langle Rel' \rangle} R$
and $Rel \circ Rel' \subseteq Rel''$
and $\bigwedge S T. (S, T) \in Rel \Longrightarrow S \rightsquigarrow \hat{\langle Rel \rangle} T$

shows $P \rightsquigarrow \hat{\langle Rel'' \rangle} R$
<proof>

lemma *weakMonotonic*:

fixes $P :: ccs$
and $A :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $B :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow \hat{\langle A \rangle} Q$
and $A \subseteq B$

shows $P \rightsquigarrow^{\hat{}} \langle B \rangle Q$
 ⟨proof⟩

lemma *simWeakSim*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $P \rightsquigarrow[Rel] Q$

shows $P \rightsquigarrow^{\hat{}} \langle Rel \rangle Q$
 ⟨proof⟩

end

theory *Weak-Cong-Sim*
imports *Weak-Cong-Semantics Weak-Sim Strong-Sim*
begin

definition *weakCongSimulation* :: $ccs \Rightarrow (ccs \times ccs) \text{ set} \Rightarrow ccs \Rightarrow \text{bool}$ ($\langle \cdot \rightsquigarrow \langle \cdot \rangle \cdot \rangle$
 $\rightarrow [80, 80, 80] 80$)

where

$P \rightsquigarrow \langle Rel \rangle Q \equiv \forall a Q'. Q \mapsto a \prec Q' \longrightarrow (\exists P'. P \Longrightarrow a \prec P' \wedge (P', Q') \in Rel)$

lemma *weakSimI*[*case-names Sim*]:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $\bigwedge \alpha Q'. Q \mapsto \alpha \prec Q' \Longrightarrow \exists P'. P \Longrightarrow \alpha \prec P' \wedge (P', Q') \in Rel$

shows $P \rightsquigarrow \langle Rel \rangle Q$
 ⟨proof⟩

lemma *weakSimE*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $\alpha :: \text{act}$
and $Q' :: ccs$

assumes $P \rightsquigarrow \langle Rel \rangle Q$
and $Q \mapsto \alpha \prec Q'$

obtains P' **where** $P \Longrightarrow \alpha \prec P'$ **and** $(P', Q') \in Rel$
 ⟨proof⟩

lemma *simWeakSim*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $P \rightsquigarrow[Rel] Q$

shows $P \rightsquigarrow\langle Rel \rangle Q$
 $\langle proof \rangle$

lemma *weakCongSimWeakSim*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $P \rightsquigarrow\langle Rel \rangle Q$

shows $P \rightsquigarrow^{\wedge}\langle Rel \rangle Q$
 $\langle proof \rangle$

lemma *test*:
assumes $P \Longrightarrow_{\tau} P'$

shows $P = P' \vee (\exists P''. P \mapsto_{\tau} \prec P'' \wedge P'' \Longrightarrow_{\tau} P')$
 $\langle proof \rangle$

lemma *tauChainCasesSym*[*consumes 1, case-names cTauNil cTauStep*]:
assumes $P \Longrightarrow_{\tau} P'$
and $Prop P$
and $\bigwedge P''. \llbracket P \mapsto_{\tau} \prec P''; P'' \Longrightarrow_{\tau} P' \rrbracket \Longrightarrow Prop P'$

shows $Prop P'$
 $\langle proof \rangle$

lemma *simE2*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $\alpha :: act$
and $Q' :: ccs$

assumes $P \rightsquigarrow\langle Rel \rangle Q$
and $Q \Longrightarrow_{\alpha} \prec Q'$
and $Sim: \bigwedge R S. (R, S) \in Rel \Longrightarrow R \rightsquigarrow^{\wedge}\langle Rel \rangle S$

obtains P' **where** $P \Longrightarrow_{\alpha} \prec P'$ **and** $(P', Q') \in Rel$
 $\langle proof \rangle$

lemma *reflexive*:
fixes $P :: ccs$

```

and Rel :: (ccs × ccs) set

assumes Id ⊆ Rel

shows P ~><Rel> P
⟨proof⟩

lemma transitive:
fixes P :: ccs
and Rel :: (ccs × ccs) set
and Q :: ccs
and Rel' :: (ccs × ccs) set
and R :: ccs
and Rel'' :: (ccs × ccs) set

assumes P ~><Rel> Q
and Q ~><Rel'> R
and Rel ∘ Rel' ⊆ Rel''
and ⋀ S T. (S, T) ∈ Rel ⇒ S ~>^<Rel> T

shows P ~><Rel''> R
⟨proof⟩

lemma weakMonotonic:
fixes P :: ccs
and A :: (ccs × ccs) set
and Q :: ccs
and B :: (ccs × ccs) set

assumes P ~><A> Q
and A ⊆ B

shows P ~><B> Q
⟨proof⟩

end

theory Strong-Sim-SC
imports Strong-Sim
begin

lemma resNilLeft:
fixes x :: name

shows (νx)0 ~>[Rel] 0
⟨proof⟩

lemma resNilRight:
fixes x :: name

```

shows $\mathbf{0} \rightsquigarrow_{[Rel]} (\nu x)\mathbf{0}$
 $\langle proof \rangle$

lemma *test[simp]*:
fixes $x :: name$
and $P :: ccs$

shows $x \# [x].P$
 $\langle proof \rangle$

lemma *scopeExtSumLeft*:
fixes $x :: name$
and $P :: ccs$
and $Q :: ccs$

assumes $x \# P$
and $C1: \bigwedge y R. y \# R \implies ((\nu y)R, R) \in Rel$
and $Id \subseteq Rel$

shows $(\nu x)(P \oplus Q) \rightsquigarrow_{[Rel]} P \oplus (\nu x)Q$
 $\langle proof \rangle$

lemma *scopeExtSumRight*:
fixes $x :: name$
and $P :: ccs$
and $Q :: ccs$

assumes $x \# P$
and $C1: \bigwedge y R. y \# R \implies (R, (\nu y)R) \in Rel$
and $Id \subseteq Rel$

shows $P \oplus (\nu x)Q \rightsquigarrow_{[Rel]} (\nu x)(P \oplus Q)$
 $\langle proof \rangle$

lemma *scopeExtLeft*:
fixes $x :: name$
and $P :: ccs$
and $Q :: ccs$

assumes $x \# P$
and $C1: \bigwedge y R T. y \# R \implies ((\nu y)(R \parallel T), R \parallel (\nu y)T) \in Rel$

shows $(\nu x)(P \parallel Q) \rightsquigarrow_{[Rel]} P \parallel (\nu x)Q$
 $\langle proof \rangle$

lemma *scopeExtRight*:
fixes $x :: name$
and $P :: ccs$

and $Q :: ccs$

assumes $x \# P$

and $CI: \bigwedge y R T. y \# R \implies (R \parallel (\nu y)T, (\nu y)(R \parallel T)) \in Rel$

shows $P \parallel (\nu x)Q \rightsquigarrow[Rel] (\nu x)(P \parallel Q)$

<proof>

lemma *sumComm*:

fixes $P :: ccs$

and $Q :: ccs$

assumes $Id \subseteq Rel$

shows $P \oplus Q \rightsquigarrow[Rel] Q \oplus P$

<proof>

lemma *sumAssocLeft*:

fixes $P :: ccs$

and $Q :: ccs$

and $R :: ccs$

assumes $Id \subseteq Rel$

shows $(P \oplus Q) \oplus R \rightsquigarrow[Rel] P \oplus (Q \oplus R)$

<proof>

lemma *sumAssocRight*:

fixes $P :: ccs$

and $Q :: ccs$

and $R :: ccs$

assumes $Id \subseteq Rel$

shows $P \oplus (Q \oplus R) \rightsquigarrow[Rel] (P \oplus Q) \oplus R$

<proof>

lemma *sumIdLeft*:

fixes $P :: ccs$

and $Rel :: (ccs \times ccs) \text{ set}$

assumes $Id \subseteq Rel$

shows $P \oplus \mathbf{0} \rightsquigarrow[Rel] P$

<proof>

lemma *sumIdRight*:

fixes $P :: ccs$

and $Rel :: (ccs \times ccs) \text{ set}$

assumes $Id \subseteq Rel$

shows $P \rightsquigarrow[Rel] P \oplus \mathbf{0}$
<proof>

lemma *parComm*:

fixes $P :: ccs$
and $Q :: ccs$

assumes $C1: \bigwedge R T. (R \parallel T, T \parallel R) \in Rel$

shows $P \parallel Q \rightsquigarrow[Rel] Q \parallel P$
<proof>

lemma *parAssocLeft*:

fixes $P :: ccs$
and $Q :: ccs$
and $R :: ccs$

assumes $C1: \bigwedge S T U. ((S \parallel T) \parallel U, S \parallel (T \parallel U)) \in Rel$

shows $(P \parallel Q) \parallel R \rightsquigarrow[Rel] P \parallel (Q \parallel R)$
<proof>

lemma *parAssocRight*:

fixes $P :: ccs$
and $Q :: ccs$
and $R :: ccs$

assumes $C1: \bigwedge S T U. (S \parallel (T \parallel U), (S \parallel T) \parallel U) \in Rel$

shows $P \parallel (Q \parallel R) \rightsquigarrow[Rel] (P \parallel Q) \parallel R$
<proof>

lemma *parIdLeft*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $\bigwedge Q. (Q \parallel \mathbf{0}, Q) \in Rel$

shows $P \parallel \mathbf{0} \rightsquigarrow[Rel] P$
<proof>

lemma *parIdRight*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $\bigwedge Q. (Q, Q \parallel \mathbf{0}) \in Rel$

shows $P \rightsquigarrow[Rel] P \parallel \mathbf{0}$
 $\langle proof \rangle$

declare *fresh-atm*[*simp*]

lemma *resActLeft*:

fixes $x :: name$
and $\alpha :: act$
and $P :: ccs$

assumes $x \# \alpha$
and $Id \subseteq Rel$

shows $(\nu x)(\alpha.(P)) \rightsquigarrow[Rel] (\alpha.(\nu x)P)$
 $\langle proof \rangle$

lemma *resActRight*:

fixes $x :: name$
and $\alpha :: act$
and $P :: ccs$

assumes $x \# \alpha$
and $Id \subseteq Rel$

shows $\alpha.(\nu x)P \rightsquigarrow[Rel] (\nu x)(\alpha.(P))$
 $\langle proof \rangle$

lemma *resComm*:

fixes $x :: name$
and $y :: name$
and $P :: ccs$

assumes $\bigwedge Q. ((\nu x)((\nu y)Q), (\nu y)((\nu x)Q)) \in Rel$

shows $(\nu x)((\nu y)P) \rightsquigarrow[Rel] (\nu y)((\nu x)P)$
 $\langle proof \rangle$

inductive-cases *bangCases*[*simplified ccs.distinct act.distinct*]: $!P \mapsto \alpha \prec P'$

lemma *bangUnfoldLeft*:

fixes $P :: ccs$

assumes $Id \subseteq Rel$

shows $P \parallel !P \rightsquigarrow[Rel] !P$
 $\langle proof \rangle$

lemma *bangUnfoldRight*:

```

fixes P :: ccs

assumes Id  $\subseteq$  Rel

shows !P  $\rightsquigarrow$ [Rel] P || !P
<proof>

end

theory Strong-Bisim
  imports Strong-Sim
begin

lemma monotonic:
  fixes P :: ccs
  and A :: (ccs  $\times$  ccs) set
  and Q :: ccs
  and B :: (ccs  $\times$  ccs) set

  assumes P  $\rightsquigarrow$ [A] Q
  and A  $\subseteq$  B

  shows P  $\rightsquigarrow$ [B] Q
  <proof>

lemma monoCoinduct:  $\bigwedge x y xa xb P Q.$ 
  
$$x \leq y \implies$$

  
$$(Q \rightsquigarrow[\{(xb, xa). x xb xa\}] P) \longrightarrow$$

  
$$(Q \rightsquigarrow[\{(xb, xa). y xb xa\}] P)$$

  <proof>

coinductive-set bisim :: (ccs  $\times$  ccs) set
where
  
$$\llbracket P \rightsquigarrow[bisim] Q; (Q, P) \in bisim \rrbracket \implies (P, Q) \in bisim$$

monos monoCoinduct

abbreviation
  bisimJudge ( $\leftarrow \sim \rightarrow$  [70, 70] 65) where P  $\sim$  Q  $\equiv$  (P, Q)  $\in$  bisim

lemma bisimCoinductAux[consumes 1]:
  fixes P :: ccs
  and Q :: ccs
  and X :: (ccs  $\times$  ccs) set

  assumes (P, Q)  $\in$  X
  and  $\bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow[(X \cup bisim)] Q \wedge (Q, P) \in X$ 

  shows P  $\sim$  Q
  <proof>

```

lemma *bisimCoinduct*[*consumes 1*, *case-names cSim cSym*]:

fixes $P :: ccs$
and $Q :: ccs$
and $X :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in X$
and $\bigwedge R S. (R, S) \in X \implies R \rightsquigarrow[(X \cup \text{bisim})] S$
and $\bigwedge R S. (R, S) \in X \implies (S, R) \in X$

shows $P \sim Q$
(*proof*)

lemma *bisimWeakCoinductAux*[*consumes 1*]:

fixes $P :: ccs$
and $Q :: ccs$
and $X :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in X$
and $\bigwedge R S. (R, S) \in X \implies R \rightsquigarrow[X] S \wedge (S, R) \in X$

shows $P \sim Q$
(*proof*)

lemma *bisimWeakCoinduct*[*consumes 1*, *case-names cSim cSym*]:

fixes $P :: ccs$
and $Q :: ccs$
and $X :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in X$
and $\bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow[X] Q$
and $\bigwedge P Q. (P, Q) \in X \implies (Q, P) \in X$

shows $P \sim Q$
(*proof*)

lemma *bisimE*:

fixes $P :: ccs$
and $Q :: ccs$

assumes $P \sim Q$

shows $P \rightsquigarrow[\text{bisim}] Q$
and $Q \sim P$
(*proof*)

lemma *bisimI*:

fixes $P :: ccs$
and $Q :: ccs$

```

assumes  $P \rightsquigarrow[bisim] Q$ 
and  $Q \sim P$ 

shows  $P \sim Q$ 
⟨proof⟩

lemma reflexive:
fixes  $P :: ccs$ 

shows  $P \sim P$ 
⟨proof⟩

lemma symmetric:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 

assumes  $P \sim Q$ 

shows  $Q \sim P$ 
⟨proof⟩

lemma transitive:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $R :: ccs$ 

assumes  $P \sim Q$ 
and  $Q \sim R$ 

shows  $P \sim R$ 
⟨proof⟩

lemma bisimTransCoinduct[consumes 1, case-names cSim cSym]:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 

assumes  $(P, Q) \in X$ 
and  $rSim: \bigwedge R S. (R, S) \in X \implies R \rightsquigarrow[(bisim\ O\ X\ O\ bisim)] S$ 
and  $rSym: \bigwedge R S. (R, S) \in X \implies (S, R) \in X$ 

shows  $P \sim Q$ 
⟨proof⟩

end

theory Strong-Sim-Pres
imports Strong-Sim
begin

```

lemma *actPres*:

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $a :: name$
and $Rel' :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in Rel$

shows $\alpha.(P) \rightsquigarrow[Rel] \alpha.(Q)$
<proof>

lemma *sumPres*:

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow[Rel] Q$
and $Rel \subseteq Rel'$
and $Id \subseteq Rel'$

shows $P \oplus R \rightsquigarrow[Rel'] Q \oplus R$
<proof>

lemma *parPresAux*:

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow[Rel] Q$
and $(P, Q) \in Rel$
and $R \rightsquigarrow[Rel'] T$
and $(R, T) \in Rel'$
and $C1: \bigwedge P' Q' R' T'. \llbracket (P', Q') \in Rel; (R', T') \in Rel' \rrbracket \implies (P' \parallel R', Q' \parallel T') \in Rel''$

shows $P \parallel R \rightsquigarrow[Rel''] Q \parallel T$
<proof>

lemma *parPres*:

fixes $P :: ccs$
and $Q :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow[Rel] Q$
and $(P, Q) \in Rel$
and $C1: \bigwedge S T U. (S, T) \in Rel \implies (S \parallel U, T \parallel U) \in Rel'$

shows $P \parallel R \rightsquigarrow[Rel'] Q \parallel R$
<proof>

lemma *resPres*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$
and $x :: name$

assumes $P \rightsquigarrow[Rel] Q$
and $\bigwedge R S y. (R, S) \in Rel \implies ((\nu y)R, (\nu y)S) \in Rel'$

shows $(\nu x)P \rightsquigarrow[Rel'] (\nu x)Q$
<proof>

lemma *bangPres*:

fixes $P :: ccs$
and $Rel :: (ccs \times ccs) \text{ set}$
and $Q :: ccs$

assumes $(P, Q) \in Rel$
and $C1: \bigwedge R S. (R, S) \in Rel \implies R \rightsquigarrow[Rel] S$

shows $!P \rightsquigarrow[bangRel\ Rel] !Q$
<proof>

end

theory *Strong-Bisim-Pres*

imports *Strong-Bisim Strong-Sim-Pres*
begin

lemma *actPres*:

fixes $P :: ccs$
and $Q :: ccs$
and $\alpha :: act$

assumes $P \sim Q$

shows $\alpha.(P) \sim \alpha.(Q)$
<proof>

lemma *sumPres*:

fixes $P :: ccs$
and $Q :: ccs$
and $R :: ccs$

assumes $P \sim Q$

shows $P \oplus R \sim Q \oplus R$
 ⟨*proof*⟩

lemma *parPres*:
fixes $P :: ccs$
and $Q :: ccs$
and $R :: ccs$

assumes $P \sim Q$

shows $P \parallel R \sim Q \parallel R$
 ⟨*proof*⟩

lemma *resPres*:
fixes $P :: ccs$
and $Q :: ccs$
and $x :: name$

assumes $P \sim Q$

shows $(\nu x)P \sim (\nu x)Q$
 ⟨*proof*⟩

lemma *bangPres*:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \sim Q$

shows $!P \sim !Q$
 ⟨*proof*⟩

end

theory *Struct-Cong*
imports *Agent*
begin

inductive *structCong* :: $ccs \Rightarrow ccs \Rightarrow bool$ ($\langle - \equiv_s - \rangle$)

where

Refl: $P \equiv_s P$

| *Sym*: $P \equiv_s Q \Longrightarrow Q \equiv_s P$

| *Trans*: $\llbracket P \equiv_s Q; Q \equiv_s R \rrbracket \Longrightarrow P \equiv_s R$

| *ParComm*: $P \parallel Q \equiv_s Q \parallel P$

| *ParAssoc*: $(P \parallel Q) \parallel R \equiv_s P \parallel (Q \parallel R)$

| *ParId*: $P \parallel \mathbf{0} \equiv_s P$

| *SumComm*: $P \oplus Q \equiv_s Q \oplus P$

| *SumAssoc*: $(P \oplus Q) \oplus R \equiv_s P \oplus (Q \oplus R)$
| *SumId*: $P \oplus \mathbf{0} \equiv_s P$

| *ResNil*: $(\nu x)\mathbf{0} \equiv_s \mathbf{0}$
| *ScopeExtPar*: $x \# P \implies (\nu x)(P \parallel Q) \equiv_s P \parallel (\nu x)Q$
| *ScopeExtSum*: $x \# P \implies (\nu x)(P \oplus Q) \equiv_s P \oplus (\nu x)Q$
| *ScopeAct*: $x \# \alpha \implies (\nu x)(\alpha.(P)) \equiv_s \alpha.(\nu x)P$
| *ScopeCommAux*: $x \neq y \implies (\nu x)((\nu y)P) \equiv_s (\nu y)((\nu x)P)$

| *BangUnfold*: $!P \equiv_s P \parallel !P$
equivariance *structCong*
nominal-inductive *structCong*
 $\langle proof \rangle$

lemma *ScopeComm*:
fixes $x :: name$
and $y :: name$
and $P :: ccs$

shows $(\nu x)((\nu y)P) \equiv_s (\nu y)((\nu x)P)$
 $\langle proof \rangle$

end

theory *Strong-Bisim-SC*
imports *Strong-Sim-SC Strong-Bisim-Pres Struct-Cong*
begin

lemma *resNil*:
fixes $x :: name$

shows $(\nu x)\mathbf{0} \sim \mathbf{0}$
 $\langle proof \rangle$

lemma *scopeExt*:
fixes $x :: name$
and $P :: ccs$
and $Q :: ccs$

assumes $x \# P$

shows $(\nu x)(P \parallel Q) \sim P \parallel (\nu x)Q$
 $\langle proof \rangle$

lemma *sumComm*:
fixes $P :: ccs$
and $Q :: ccs$

shows $P \oplus Q \sim Q \oplus P$

$\langle proof \rangle$

lemma *sumAssoc*:

fixes $P :: ccs$

and $Q :: ccs$

and $R :: ccs$

shows $(P \oplus Q) \oplus R \sim P \oplus (Q \oplus R)$

$\langle proof \rangle$

lemma *sumId*:

fixes $P :: ccs$

shows $P \oplus \mathbf{0} \sim P$

$\langle proof \rangle$

lemma *parComm*:

fixes $P :: ccs$

and $Q :: ccs$

shows $P \parallel Q \sim Q \parallel P$

$\langle proof \rangle$

lemma *parAssoc*:

fixes $P :: ccs$

and $Q :: ccs$

and $R :: ccs$

shows $(P \parallel Q) \parallel R \sim P \parallel (Q \parallel R)$

$\langle proof \rangle$

lemma *parId*:

fixes $P :: ccs$

shows $P \parallel \mathbf{0} \sim P$

$\langle proof \rangle$

lemma *scopeFresh*:

fixes $x :: name$

and $P :: ccs$

assumes $x \nmid P$

shows $(\nu x)P \sim P$

$\langle proof \rangle$

lemma *scopeExtSum*:

fixes $x :: name$

and $P :: ccs$

```

and Q :: ccs

assumes x # P

shows ( $\nu x$ ).(P  $\oplus$  Q)  $\sim$  P  $\oplus$  ( $\nu x$ ).Q
<proof>

lemma resAct:
fixes x :: name
and  $\alpha$  :: act
and P :: ccs

assumes x #  $\alpha$ 

shows ( $\nu x$ ).( $\alpha$ .(P))  $\sim$   $\alpha$ .(( $\nu x$ ).P)
<proof>

lemma resComm:
fixes x :: name
and y :: name
and P :: ccs

shows ( $\nu x$ ).( ( $\nu y$ ).P )  $\sim$  ( $\nu y$ ).( ( $\nu x$ ).P )
<proof>

lemma bangUnfold:
fixes P

shows !P  $\sim$  P || !P
<proof>

lemma bisimStructCong:
fixes P :: ccs
and Q :: ccs

assumes P  $\equiv_s$  Q

shows P  $\sim$  Q
<proof>

end

theory Weak-Bisim
imports Weak-Sim Strong-Bisim-SC Struct-Cong
begin

lemma weakMonoCoinduct:  $\bigwedge x y xa xb P Q.$ 

$$x \leq y \implies (Q \rightsquigarrow^* \langle \{(xb, xa). x xb xa \} \rangle P) \longrightarrow$$


```

$(Q \rightsquigarrow^{\hat{\cdot}} \langle \{(xb, xa). y\} \rangle P)$

<proof>

coinductive-set *weakBisimulation* :: (ccs × ccs) set

where

$\llbracket P \rightsquigarrow^{\hat{\cdot}} \langle \text{weakBisimulation} \rangle Q; (Q, P) \in \text{weakBisimulation} \rrbracket \implies (P, Q) \in \text{weakBisimulation}$

monos *weakMonoCoinduct*

abbreviation

weakBisimJudge ($\langle \cdot \approx \cdot \rangle$ [70, 70] 65) **where** $P \approx Q \equiv (P, Q) \in \text{weakBisimulation}$

lemma *weakBisimulationCoinductAux*[*consumes 1*]:

fixes $P :: \text{ccs}$

and $Q :: \text{ccs}$

and $X :: (\text{ccs} \times \text{ccs}) \text{ set}$

assumes $(P, Q) \in X$

and $\bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow^{\hat{\cdot}} \langle (X \cup \text{weakBisimulation}) \rangle Q \wedge (Q, P) \in X$

shows $P \approx Q$

<proof>

lemma *weakBisimulationCoinduct*[*consumes 1*, *case-names cSim cSym*]:

fixes $P :: \text{ccs}$

and $Q :: \text{ccs}$

and $X :: (\text{ccs} \times \text{ccs}) \text{ set}$

assumes $(P, Q) \in X$

and $\bigwedge R S. (R, S) \in X \implies R \rightsquigarrow^{\hat{\cdot}} \langle (X \cup \text{weakBisimulation}) \rangle S$

and $\bigwedge R S. (R, S) \in X \implies (S, R) \in X$

shows $P \approx Q$

<proof>

lemma *weakBisimWeakCoinductAux*[*consumes 1*]:

fixes $P :: \text{ccs}$

and $Q :: \text{ccs}$

and $X :: (\text{ccs} \times \text{ccs}) \text{ set}$

assumes $(P, Q) \in X$

and $\bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow^{\hat{\cdot}} \langle X \rangle Q \wedge (Q, P) \in X$

shows $P \approx Q$

<proof>

lemma *weakBisimWeakCoinduct*[*consumes 1*, *case-names cSim cSym*]:

fixes $P :: \text{ccs}$

```

and    $Q :: ccs$ 
and    $X :: (ccs \times ccs) \text{ set}$ 

assumes  $(P, Q) \in X$ 
and      $\bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow^{\langle X \rangle} Q$ 
and      $\bigwedge P Q. (P, Q) \in X \implies (Q, P) \in X$ 

shows  $P \approx Q$ 
<proof>

lemma weakBisimulationE:
  fixes  $P :: ccs$ 
  and    $Q :: ccs$ 

  assumes  $P \approx Q$ 

  shows  $P \rightsquigarrow^{\langle \text{weakBisimulation} \rangle} Q$ 
  and    $Q \approx P$ 
<proof>

lemma weakBisimulationI:
  fixes  $P :: ccs$ 
  and    $Q :: ccs$ 

  assumes  $P \rightsquigarrow^{\langle \text{weakBisimulation} \rangle} Q$ 
  and      $Q \approx P$ 

  shows  $P \approx Q$ 
<proof>

lemma reflexive:
  fixes  $P :: ccs$ 

  shows  $P \approx P$ 
<proof>

lemma symmetric:
  fixes  $P :: ccs$ 
  and    $Q :: ccs$ 

  assumes  $P \approx Q$ 

  shows  $Q \approx P$ 
<proof>

lemma transitive:
  fixes  $P :: ccs$ 
  and    $Q :: ccs$ 
  and    $R :: ccs$ 

```

assumes $P \approx Q$
and $Q \approx R$

shows $P \approx R$
 $\langle proof \rangle$

lemma *bisimWeakBisimulation*:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \sim Q$

shows $P \approx Q$
 $\langle proof \rangle$

lemma *structCongWeakBisimulation*:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \equiv_s Q$

shows $P \approx Q$
 $\langle proof \rangle$

lemma *strongAppend*:
fixes $P \quad :: ccs$
and $Q \quad :: ccs$
and $R \quad :: ccs$
and $Rel \quad :: (ccs \times ccs) \text{ set}$
and $Rel' \quad :: (ccs \times ccs) \text{ set}$
and $Rel'' \quad :: (ccs \times ccs) \text{ set}$

assumes $PSimQ: P \rightsquigarrow^{\wedge} \langle Rel \rangle Q$
and $QSimR: Q \rightsquigarrow [Rel'] R$
and $Trans: Rel \circ Rel' \subseteq Rel''$

shows $P \rightsquigarrow^{\wedge} \langle Rel'' \rangle R$
 $\langle proof \rangle$

lemma *weakBisimWeakUpto*[*case-names cSim cSym, consumes 1*]:
assumes $p: (P, Q) \in X$
and $rSim: \bigwedge P Q. (P, Q) \in X \implies P \rightsquigarrow^{\wedge} \langle (weakBisimulation \circ X \circ bisim) \rangle Q$
and $rSym: \bigwedge P Q. (P, Q) \in X \implies (Q, P) \in X$

shows $P \approx Q$
 $\langle proof \rangle$

lemma *weakBisimUpto*[*case-names cSim cSym, consumes 1*]:

```

assumes  $p: (P, Q) \in X$ 
and  $rSim: \bigwedge R S. (R, S) \in X \implies R \rightsquigarrow^{\langle weakBisimulation\ O\ (X \cup weakBisimulation)\ O\ bisim \rangle} S$ 
and  $rSym: \bigwedge R S. (R, S) \in X \implies (S, R) \in X$ 

shows  $P \approx Q$ 
 $\langle proof \rangle$ 

```

end

theory *Weak-Cong*

imports *Weak-Cong-Sim Weak-Bisim Strong-Bisim-SC*

begin

definition *weakCongruence* :: $ccs \Rightarrow ccs \Rightarrow bool$ ($\langle - \cong - \rangle$ [70, 70] 65)

where

$P \cong Q \equiv P \rightsquigarrow^{\langle weakBisimulation \rangle} Q \wedge Q \rightsquigarrow^{\langle weakBisimulation \rangle} P$

lemma *weakCongruenceE*:

fixes $P :: ccs$

and $Q :: ccs$

assumes $P \cong Q$

shows $P \rightsquigarrow^{\langle weakBisimulation \rangle} Q$

and $Q \rightsquigarrow^{\langle weakBisimulation \rangle} P$

$\langle proof \rangle$

lemma *weakCongruenceI*:

fixes $P :: ccs$

and $Q :: ccs$

assumes $P \rightsquigarrow^{\langle weakBisimulation \rangle} Q$

and $Q \rightsquigarrow^{\langle weakBisimulation \rangle} P$

shows $P \cong Q$

$\langle proof \rangle$

lemma *weakCongISym*[*consumes 1, case-names cSym cSim*]:

fixes $P :: ccs$

and $Q :: ccs$

assumes *Prop* $P\ Q$

and $\bigwedge P\ Q. Prop\ P\ Q \implies Prop\ Q\ P$

and $\bigwedge P\ Q. Prop\ P\ Q \implies (F\ P) \rightsquigarrow^{\langle weakBisimulation \rangle} (F\ Q)$

shows $F\ P \cong F\ Q$

$\langle proof \rangle$

lemma *weakCongISym2*[*consumes 1, case-names cSim*]:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \cong Q$
and $\bigwedge P Q. P \cong Q \implies (F P) \rightsquigarrow \langle \text{weakBisimulation} \rangle (F Q)$

shows $F P \cong F Q$
 $\langle \text{proof} \rangle$

lemma *reflexive*:
fixes $P :: ccs$

shows $P \cong P$
 $\langle \text{proof} \rangle$

lemma *symmetric*:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \cong Q$

shows $Q \cong P$
 $\langle \text{proof} \rangle$

lemma *transitive*:
fixes $P :: ccs$
and $Q :: ccs$
and $R :: ccs$

assumes $P \cong Q$
and $Q \cong R$

shows $P \cong R$
 $\langle \text{proof} \rangle$

lemma *bisimWeakCongruence*:
fixes $P :: ccs$
and $Q :: ccs$

assumes $P \sim Q$

shows $P \cong Q$
 $\langle \text{proof} \rangle$

lemma *structCongWeakCongruence*:
fixes $P :: ccs$
and $Q :: ccs$

```

assumes  $P \equiv_s Q$ 

shows  $P \cong Q$ 
⟨proof⟩

lemma weakCongruenceWeakBisimulation:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 

assumes  $P \cong Q$ 

shows  $P \approx Q$ 
⟨proof⟩

end

theory Weak-Sim-Pres
imports Weak-Sim
begin

lemma actPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $Rel :: (ccs \times ccs) \text{ set}$ 
and  $a :: \text{name}$ 
and  $Rel' :: (ccs \times ccs) \text{ set}$ 

assumes  $(P, Q) \in Rel$ 

shows  $\alpha.(P) \rightsquigarrow^{\langle Rel \rangle} \alpha.(Q)$ 
⟨proof⟩

lemma sumPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $Rel :: (ccs \times ccs) \text{ set}$ 

assumes  $P \rightsquigarrow^{\langle Rel \rangle} Q$ 
and  $Rel \subseteq Rel'$ 
and  $Id \subseteq Rel'$ 
and  $C1: \bigwedge S T U. (S, T) \in Rel \implies (S \oplus U, T) \in Rel'$ 

shows  $P \oplus R \rightsquigarrow^{\langle Rel' \rangle} Q \oplus R$ 
⟨proof⟩

lemma parPresAux:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 

```

and $R \quad :: \text{ccs}$
and $T \quad :: \text{ccs}$
and $Rel \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
and $Rel' \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
and $Rel'' \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$

assumes $P \rightsquigarrow^{\hat{}} \langle Rel \rangle Q$
and $(P, Q) \in Rel$
and $R \rightsquigarrow^{\hat{}} \langle Rel' \rangle T$
and $(R, T) \in Rel'$
and $C1: \bigwedge P' Q' R' T'. \llbracket (P', Q') \in Rel; (R', T') \in Rel' \rrbracket \implies (P' \parallel R', Q' \parallel T') \in Rel''$

shows $P \parallel R \rightsquigarrow^{\hat{}} \langle Rel'' \rangle Q \parallel T$
 $\langle \text{proof} \rangle$

lemma *parPres*:
fixes $P \quad :: \text{ccs}$
and $Q \quad :: \text{ccs}$
and $R \quad :: \text{ccs}$
and $Rel \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
and $Rel' \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
assumes $P \rightsquigarrow^{\hat{}} \langle Rel \rangle Q$
and $(P, Q) \in Rel$
and $C1: \bigwedge S T U. (S, T) \in Rel \implies (S \parallel U, T \parallel U) \in Rel'$

shows $P \parallel R \rightsquigarrow^{\hat{}} \langle Rel' \rangle Q \parallel R$
 $\langle \text{proof} \rangle$

lemma *resPres*:
fixes $P \quad :: \text{ccs}$
and $Rel \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
and $Q \quad :: \text{ccs}$
and $x \quad :: \text{name}$

assumes $P \rightsquigarrow^{\hat{}} \langle Rel \rangle Q$
and $\bigwedge R S y. (R, S) \in Rel \implies ((\nu y)R, (\nu y)S) \in Rel'$

shows $(\nu x)P \rightsquigarrow^{\hat{}} \langle Rel' \rangle (\nu x)Q$
 $\langle \text{proof} \rangle$

lemma *bangPres*:
fixes $P \quad :: \text{ccs}$
and $Rel \quad :: (\text{ccs} \times \text{ccs}) \text{ set}$
and $Q \quad :: \text{ccs}$

assumes $(P, Q) \in Rel$
and $C1: \bigwedge R S. (R, S) \in Rel \implies R \rightsquigarrow^{\hat{}} \langle Rel \rangle S$
and $Par: \bigwedge R S T U. \llbracket (R, S) \in Rel; (T, U) \in Rel \rrbracket \implies (R \parallel T, S \parallel U) \in Rel$

```

Rel'
  and C2: bangRel Rel  $\subseteq$  Rel'
  and C3:  $\bigwedge R S. (R \parallel !R, S) \in Rel' \implies (!R, S) \in Rel'$ 

  shows  $!P \rightsquigarrow^{\hat{}} \langle Rel' \rangle !Q$ 
  <proof>

end

theory Weak-Bisim-Pres
  imports Weak-Bisim Weak-Sim-Pres Strong-Bisim-SC
begin

lemma actPres:
  fixes P :: ccs
  and Q :: ccs
  and  $\alpha$  :: act

  assumes  $P \approx Q$ 

  shows  $\alpha.(P) \approx \alpha.(Q)$ 
  <proof>

lemma parPres:
  fixes P :: ccs
  and Q :: ccs
  and R :: ccs

  assumes  $P \approx Q$ 

  shows  $P \parallel R \approx Q \parallel R$ 
  <proof>

lemma resPres:
  fixes P :: ccs
  and Q :: ccs
  and x :: name

  assumes  $P \approx Q$ 

  shows  $(\nu x)P \approx (\nu x)Q$ 
  <proof>

lemma bangPres:
  fixes P :: ccs
  and Q :: ccs

  assumes  $P \approx Q$ 

```

shows $!P \approx !Q$
<proof>

end

theory *Weak-Cong-Sim-Pres*
 imports *Weak-Cong-Sim*
begin

lemma *actPres*:
 fixes $P :: ccs$
 and $Q :: ccs$
 and $Rel :: (ccs \times ccs) \text{ set}$
 and $a :: name$
 and $Rel' :: (ccs \times ccs) \text{ set}$

assumes $(P, Q) \in Rel$

shows $\alpha.(P) \rightsquigarrow \langle Rel \rangle \alpha.(Q)$
<proof>

lemma *sumPres*:
 fixes $P :: ccs$
 and $Q :: ccs$
 and $Rel :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow \langle Rel \rangle Q$
 and $Rel \subseteq Rel'$
 and $Id \subseteq Rel'$

shows $P \oplus R \rightsquigarrow \langle Rel' \rangle Q \oplus R$
<proof>

lemma *parPres*:
 fixes $P :: ccs$
 and $Q :: ccs$
 and $Rel :: (ccs \times ccs) \text{ set}$

assumes $P \rightsquigarrow \langle Rel \rangle Q$
 and $(P, Q) \in Rel$
 and $C1: \bigwedge S T U. (S, T) \in Rel \implies (S \parallel U, T \parallel U) \in Rel'$

shows $P \parallel R \rightsquigarrow \langle Rel' \rangle Q \parallel R$
<proof>

lemma *resPres*:
 fixes $P :: ccs$
 and $Rel :: (ccs \times ccs) \text{ set}$
 and $Q :: ccs$

```

and  $x :: name$ 

assumes  $P \rightsquigarrow \langle Rel \rangle Q$ 
and  $\bigwedge R S y. (R, S) \in Rel \implies ((\nu y)R, (\nu y)S) \in Rel'$ 

shows  $(\nu x)P \rightsquigarrow \langle Rel' \rangle (\nu x)Q$ 
 $\langle proof \rangle$ 

lemma bangPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $Rel :: (ccs \times ccs) set$ 
and  $Rel' :: (ccs \times ccs) set$ 

assumes  $(P, Q) \in Rel$ 
and  $C1: \bigwedge R S. (R, S) \in Rel \implies R \rightsquigarrow \langle Rel' \rangle S$ 
and  $C2: Rel \subseteq Rel'$ 

shows  $!P \rightsquigarrow \langle bangRel\ Rel' \rangle !Q$ 
 $\langle proof \rangle$ 

end

theory Weak-Cong-Pres
imports Weak-Cong Weak-Bisim-Pres Weak-Cong-Sim-Pres
begin

lemma actPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $\alpha :: act$ 

assumes  $P \cong Q$ 

shows  $\alpha.(P) \cong \alpha.(Q)$ 
 $\langle proof \rangle$ 

lemma sumPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $R :: ccs$ 

assumes  $P \cong Q$ 

shows  $P \oplus R \cong Q \oplus R$ 
 $\langle proof \rangle$ 

lemma parPres:
fixes  $P :: ccs$ 

```

```

and  $Q :: ccs$ 
and  $R :: ccs$ 

assumes  $P \cong Q$ 

shows  $P \parallel R \cong Q \parallel R$ 
 $\langle proof \rangle$ 

lemma resPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 
and  $x :: name$ 

assumes  $P \cong Q$ 

shows  $(\nu x)P \cong (\nu x)Q$ 
 $\langle proof \rangle$ 

lemma weakBisimBangRel:  $bangRel \ weakBisimulation \subseteq weakBisimulation$ 
 $\langle proof \rangle$ 

lemma bangPres:
fixes  $P :: ccs$ 
and  $Q :: ccs$ 

assumes  $P \cong Q$ 

shows  $!P \cong !Q$ 
 $\langle proof \rangle$ 

end

```

References

- [1] J. Bengtson. *Formalising process calculi*, volume 94. Uppsala Dissertations from the Faculty of Science and Technology, 2010.