The Calculus of Communicating Systems

Jesper Bengtson

September 13, 2023

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

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 $act = actAction\ name$ $\quad ((\!|-|\!)\ 100)$
$\quad | \ actCoAction\ name \quad (\langle - \rangle\ 100)$
$\quad | \ actTau \quad (\tau\ 100)$

nominal-datatype $ccs = CCSNil \quad (\mathbf{0}\ 115)$
$\quad | \ Action\ act\ ccs \quad (-.-\ [120,\ 110]\ 110)$
$\quad | \ Sum\ ccs\ ccs \quad (\mathbf{infixl} \oplus\ 90)$
$\quad | \ Par\ ccs\ ccs \quad (\mathbf{infixl} \parallel\ 85)$
$\quad | \ Res\ «name»\ ccs \quad ((\!|\nu-|\!)-\ [105,\ 100]\ 100)$
$\quad | \ Bang\ ccs \quad (!-\ [95])$

nominal-primrec $coAction :: act \Rightarrow act$
where
  $coAction\ ((\!|a|\!)) = (\langle a \rangle)$
$|\ coAction\ (\langle a \rangle) = ((\!|a|\!))$
$|\ coAction\ (\tau) = \tau$
$\langle proof \rangle$

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

  shows $(p \cdot coAction\ a) = coAction(p \cdot a)$
$\langle proof \rangle$

lemma *coActionSimps*[*simp*]:
  fixes $a :: act$

  shows $coAction(coAction\ a) = a$
  and $(coAction\ a = \tau) = (a = \tau)$
$\langle proof \rangle$

lemma *coActSimp*[*simp*]: **shows** *coAction* $\alpha \neq \tau = (\alpha \neq \tau)$ **and** $(coAction\ \alpha = \tau) = (\alpha = \tau)$
$\langle proof \rangle$

lemma *coActFresh*[*simp*]:
fixes $x :: name$
and $a :: act$

shows $x \sharp coAction\ a = x \sharp a$
$\langle proof \rangle$

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

assumes $y \sharp P$

shows $(\!|\nu x|\!)P = (\!|\nu y|\!)([(x, y)] \cdot P)$
$\langle proof \rangle$

inductive *semantics* :: $ccs \Rightarrow act \Rightarrow ccs \Rightarrow bool$ $(\text{-} \longmapsto\text{-} \prec \text{-}\ [80, 80, 80]\ 80)$
where
Action: $\alpha.(P) \longmapsto\alpha \prec P$
| *Sum1*: $P \longmapsto\alpha \prec P' \Longrightarrow P \oplus Q \longmapsto\alpha \prec P'$
| *Sum2*: $Q \longmapsto\alpha \prec Q' \Longrightarrow P \oplus Q \longmapsto\alpha \prec Q'$
| *Par1*: $P \longmapsto\alpha \prec P' \Longrightarrow P \parallel Q \longmapsto\alpha \prec P' \parallel Q$
| *Par2*: $Q \longmapsto\alpha \prec Q' \Longrightarrow P \parallel Q \longmapsto\alpha \prec P \parallel Q'$
| *Comm*: $[\![P \longmapsto a \prec P'; Q \longmapsto(coAction\ a) \prec Q'; a \neq \tau]\!] \Longrightarrow P \parallel Q \longmapsto\tau \prec P' \parallel Q'$
| *Res*: $[\![P \longmapsto\alpha \prec P'; x \sharp \alpha]\!] \Longrightarrow (\!|\nu x|\!)P \longmapsto\alpha \prec (\!|\nu x|\!)P'$
| *Bang*: $P \parallel !P \longmapsto\alpha \prec P' \Longrightarrow !P \longmapsto\alpha \prec P'$

equivariance *semantics*

nominal-inductive *semantics*
$\langle proof \rangle$

lemma *semanticsInduct*:
$[\![R \longmapsto\beta \prec R'; \bigwedge\alpha\ P\ \mathcal{C}.\ Prop\ \mathcal{C}\ (\alpha.(P))\ \alpha\ P;$
$\bigwedge P\ \alpha\ P'\ Q\ \mathcal{C}.\ [\![P \longmapsto\alpha \prec P'; \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ P\ \alpha\ P']\!] \Longrightarrow Prop\ \mathcal{C}\ (ccs.Sum\ P\ Q)\ \alpha\ P';$
$\bigwedge Q\ \alpha\ Q'\ P\ \mathcal{C}.\ [\![Q \longmapsto\alpha \prec Q'; \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ Q\ \alpha\ Q']\!] \Longrightarrow Prop\ \mathcal{C}\ (ccs.Sum\ P\ Q)\ \alpha\ Q';$
$\bigwedge P\ \alpha\ P'\ Q\ \mathcal{C}.\ [\![P \longmapsto\alpha \prec P'; \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ P\ \alpha\ P']\!] \Longrightarrow Prop\ \mathcal{C}\ (P \parallel Q)\ \alpha\ (P' \parallel Q);$
$\bigwedge Q\ \alpha\ Q'\ P\ \mathcal{C}.\ [\![Q \longmapsto\alpha \prec Q'; \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ Q\ \alpha\ Q']\!] \Longrightarrow Prop\ \mathcal{C}\ (P \parallel Q)\ \alpha\ (P \parallel Q');$

$\bigwedge P\ a\ P'\ Q\ Q'\ \mathcal{C}.$
$[\![P \longmapsto a \prec P';\ \bigwedge\mathcal{C}.\ \mathit{Prop}\ \mathcal{C}\ P\ a\ P';\ Q \longmapsto(\mathit{coAction}\ a) \prec Q';$
$\bigwedge\mathcal{C}.\ \mathit{Prop}\ \mathcal{C}\ Q\ (\mathit{coAction}\ a)\ Q';\ a \neq \tau]\!]$
$\Longrightarrow \mathit{Prop}\ \mathcal{C}\ (P \parallel Q)\ (\tau)\ (P' \parallel Q');$
$\bigwedge P\ \alpha\ P'\ x\ \mathcal{C}.$
$[\![x \sharp \mathcal{C};\ P \longmapsto\alpha \prec P';\ \bigwedge\mathcal{C}.\ \mathit{Prop}\ \mathcal{C}\ P\ \alpha\ P';\ x \sharp \alpha]\!] \Longrightarrow \mathit{Prop}\ \mathcal{C}\ ((\!|\nu x|\!)P)\ \alpha$
$((\!|\nu x|\!)P');$
$\bigwedge P\ \alpha\ P'\ \mathcal{C}.\ [\![P \parallel !P \longmapsto\alpha \prec P';\ \bigwedge\mathcal{C}.\ \mathit{Prop}\ \mathcal{C}\ (P \parallel !P)\ \alpha\ P']\!] \Longrightarrow \mathit{Prop}\ \mathcal{C}\ !P\ \alpha\ P']\!]$

$\Longrightarrow \mathit{Prop}\ (\mathcal{C}::'a::\mathit{fs\text{-}name})\ R\ \beta\ R'$
$\langle \mathit{proof} \rangle$

lemma *NilTrans*[*dest*]:
shows $\mathbf{0} \longmapsto\alpha \prec P' \Longrightarrow \mathit{False}$
and $((\!|b|\!)).P \longmapsto\langle c\rangle \prec P' \Longrightarrow \mathit{False}$
and $((\!|b|\!)).P \longmapsto\tau \prec P' \Longrightarrow \mathit{False}$
and $(\langle b\rangle).P \longmapsto(\!|c|\!) \prec P' \Longrightarrow \mathit{False}$
and $(\langle b\rangle).P \longmapsto\tau \prec P' \Longrightarrow \mathit{False}$
$\langle \mathit{proof} \rangle$

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

assumes $P \longmapsto\alpha \prec P'$
and $x \sharp P$

shows $x \sharp \alpha$ **and** $x \sharp P'$
$\langle \mathit{proof} \rangle$

lemma *actCases*[*consumes 1*, *case-names cAct*]:
fixes α :: *act*
and P :: *ccs*
and β :: *act*
and P' :: *ccs*

assumes $\alpha.(P) \longmapsto\beta \prec P'$
and $\mathit{Prop}\ \alpha\ P$

shows $\mathit{Prop}\ \beta\ P'$
$\langle \mathit{proof} \rangle$

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

assumes $P \oplus Q \longmapsto\alpha \prec R$
and $\bigwedge P'. \; P \longmapsto\alpha \prec P' \Longrightarrow \mathit{Prop}\ P'$
and $\bigwedge Q'. \; Q \longmapsto\alpha \prec Q' \Longrightarrow \mathit{Prop}\ Q'$

shows $\mathit{Prop}\ R$
$\langle \mathit{proof} \rangle$

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

assumes $P \parallel Q \longmapsto\alpha \prec R$
and $\bigwedge P'. \; P \longmapsto\alpha \prec P' \Longrightarrow \mathit{Prop}\ \alpha\ (P' \parallel Q)$
and $\bigwedge Q'. \; Q \longmapsto\alpha \prec Q' \Longrightarrow \mathit{Prop}\ \alpha\ (P \parallel Q')$
and $\bigwedge P'\ Q'\ a. \; [\![P \longmapsto a \prec P'; \; Q \longmapsto(\mathit{coAction}\ a) \prec Q'; \; a \neq \tau; \; \alpha = \tau]\!] \Longrightarrow$ $\mathit{Prop}\ (\tau)\ (P' \parallel Q')$

shows $\mathit{Prop}\ \alpha\ R$
$\langle \mathit{proof} \rangle$

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

assumes $(\!|\nu x|\!)P \longmapsto\alpha \prec P'$
and $\bigwedge P'. \; [\![P \longmapsto\alpha \prec P'; \; x \sharp \alpha]\!] \Longrightarrow \mathit{Prop}\ ((\!|\nu x|\!)P')$

shows $\mathit{Prop}\ P'$
$\langle \mathit{proof} \rangle$

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

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

assumes $!P \longmapsto\alpha \prec P'$
and *rPar1*: $\bigwedge \alpha\ P'\ \mathcal{C}. \; [\![P \longmapsto\alpha \prec P']\!] \Longrightarrow \mathit{Prop}\ \mathcal{C}\ (P \parallel !P)\ \alpha\ (P' \parallel !P)$
and *rPar2*: $\bigwedge \alpha\ P'\ \mathcal{C}. \; [\![!P \longmapsto\alpha \prec P'; \; \bigwedge\mathcal{C}. \; \mathit{Prop}\ \mathcal{C}\ (!P)\ \alpha\ P']\!] \Longrightarrow \mathit{Prop}\ \mathcal{C}\ (P$

$\parallel !P)\ \alpha\ (P \parallel P')$
and *rComm*: $\bigwedge a\ P'\ P''\ \mathcal{C}.\ [\![P \longmapsto a \prec P';\ !P \longmapsto (coAction\ a) \prec P'';\ \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ (!P)\ (coAction\ a)\ P'';\ a \neq \tau]\!] \Longrightarrow Prop\ \mathcal{C}\ (P \parallel !P)\ (\tau)\ (P' \parallel P'')$
and *rBang*: $\bigwedge\alpha\ P'\ \mathcal{C}.\ [\![P \parallel !P \longmapsto \alpha \prec P';\ \bigwedge\mathcal{C}.\ Prop\ \mathcal{C}\ (P \parallel !P)\ \alpha\ P']\!] \Longrightarrow Prop\ \mathcal{C}\ (!P)\ \alpha\ P'$

shows $Prop\ \mathcal{C}\ (!P)\ \alpha\ P'$
⟨*proof*⟩

inductive-set *bangRel* :: $(ccs \times ccs)\ set \Rightarrow (ccs \times ccs)\ set$
for *Rel* :: $(ccs \times ccs)\ set$
where
BRBang: $(P,\ Q) \in Rel \Longrightarrow (!P,\ !Q) \in bangRel\ Rel$
| *BRPar*: $(R,\ T) \in Rel \Longrightarrow (P,\ Q) \in (bangRel\ Rel) \Longrightarrow (R \parallel P,\ T \parallel Q) \in (bangRel\ Rel)$

lemma *BRBangCases*[*consumes 1*, *case-names BRBang*]:
fixes P :: *ccs*
and Q :: *ccs*
and *Rel* :: $(ccs \times ccs)\ set$
and F :: $ccs \Rightarrow bool$

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

shows $F\ P$
⟨*proof*⟩

lemma *BRParCases*[*consumes 1*, *case-names BRPar*]:
fixes P :: *ccs*
and Q :: *ccs*
and *Rel* :: $(ccs \times ccs)\ set$
and F :: $ccs \Rightarrow bool$

assumes $(P,\ Q \parallel !Q) \in bangRel\ Rel$
and $\bigwedge P\ R.\ [\![(P,\ Q) \in Rel;\ (R,\ !Q) \in bangRel\ Rel]\!] \Longrightarrow F\ (P \parallel R)$

shows $F\ P$
⟨*proof*⟩

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

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

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

⟨*proof*⟩

end

theory *Tau-Chain*
  imports *Agent*
begin

definition *tauChain* :: *ccs* ⇒ *ccs* ⇒ *bool* (- $\Longrightarrow_\tau$ - [*80*, *80*] *80*)
  where $P \Longrightarrow_\tau P' \equiv (P, P') \in \{(P, P') \mid P\ P'.\ P \longmapsto \tau \prec P'\}\widehat{\ }*$

lemma *tauChainInduct*[*consumes 1*, *case-names Base Step*]:
  assumes $P \Longrightarrow_\tau P'$
  and *Prop P*
  and $\bigwedge P'\ P''.\ [\![P \Longrightarrow_\tau P'; P' \longmapsto \tau \prec P''; Prop\ P']\!] \Longrightarrow Prop\ P''$

  shows *Prop P′*
⟨*proof*⟩

lemma *tauChainRefl*[*simp*]:
  fixes *P* :: *ccs*

  shows $P \Longrightarrow_\tau P$
⟨*proof*⟩

lemma *tauChainCons*[*dest*]:
  fixes *P* :: *ccs*
  and *P′* :: *ccs*
  and *P″* :: *ccs*

  assumes $P \Longrightarrow_\tau P'$
  and $P' \longmapsto \tau \prec P''$

  shows $P \Longrightarrow_\tau P''$
⟨*proof*⟩

lemma *tauChainCons2*[*dest*]:
  fixes *P* :: *ccs*
  and *P′* :: *ccs*
  and *P″* :: *ccs*

  assumes $P' \longmapsto \tau \prec P''$
  and $P \Longrightarrow_\tau P'$

  shows $P \Longrightarrow_\tau P''$
⟨*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* ⇒ *act* ⇒ *ccs* ⇒ *bool* (- $\Longrightarrow$- $\prec$ - [*80*, *80*, *80*] *80*)
where $P \Longrightarrow\alpha \prec P' \equiv \exists P''\ P'''.\ P \Longrightarrow_\tau P'' \wedge P'' \longmapsto\alpha \prec P''' \wedge P''' \Longrightarrow_\tau P'$

lemma *weakCongTransE*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

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

obtains $P''\ P'''$ **where** $P \Longrightarrow_\tau P''$ **and** $P'' \longmapsto\alpha \prec P'''$ **and** $P''' \Longrightarrow_\tau P'$
⟨*proof*⟩

lemma *weakCongTransI*:
fixes P :: *ccs*
and P'' :: *ccs*
and α :: *act*
and P''' :: *ccs*
and P' :: *ccs*

assumes $P \Longrightarrow_\tau P''$
and $P'' \longmapsto \alpha \prec P'''$
and $P''' \Longrightarrow_\tau P'$

shows $P \Longrightarrow \alpha \prec P'$
$\langle proof \rangle$

lemma *transitionWeakCongTransition*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

assumes $P \longmapsto \alpha \prec P'$

shows $P \Longrightarrow \alpha \prec P'$
$\langle proof \rangle$

lemma *weakCongAction*:
fixes a :: *name*
and P :: *ccs*

shows $\alpha.(P) \Longrightarrow \alpha \prec P$
$\langle proof \rangle$

lemma *weakCongSum1*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and Q :: *ccs*

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

shows $P \oplus Q \Longrightarrow \alpha \prec P'$
$\langle proof \rangle$

lemma *weakCongSum2*:
fixes Q :: *ccs*
and α :: *act*
and Q' :: *ccs*
and P :: *ccs*

assumes $Q \Longrightarrow \alpha \prec Q'$

shows $P \oplus Q \Longrightarrow \alpha \prec Q'$
$\langle proof \rangle$

lemma *weakCongPar1*:
fixes P :: *ccs*
and α :: *act*

and P' :: *ccs*
and Q :: *ccs*

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

shows $P \parallel Q \Longrightarrow\alpha \prec P' \parallel Q$
$\langle proof \rangle$

lemma *weakCongPar2*:
fixes Q :: *ccs*
and α :: *act*
and Q' :: *ccs*
and P :: *ccs*

assumes $Q \Longrightarrow\alpha \prec Q'$

shows $P \parallel Q \Longrightarrow\alpha \prec P \parallel Q'$
$\langle proof \rangle$

lemma *weakCongSync*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and Q :: *ccs*

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

shows $P \parallel Q \Longrightarrow\tau \prec P' \parallel Q'$
$\langle proof \rangle$

lemma *weakCongRes*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and x :: *name*

assumes $P \Longrightarrow\alpha \prec P'$
and $x \sharp \alpha$

shows $(\!|\nu x|\!)P \Longrightarrow\alpha \prec (\!|\nu x|\!)P'$
$\langle proof \rangle$

lemma *weakCongRepl*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

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

shows $!P \Longrightarrow\alpha \prec P'$
$\langle proof \rangle$

end

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

definition *weakTrans* :: *ccs* ⇒ *act* ⇒ *ccs* ⇒ *bool* (- $\Longrightarrow\hat{}$ - $\prec$ - [*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 $P \Longrightarrow\hat{}\ \tau \prec P$
$\langle proof \rangle$

lemma *weakTransCases*[*consumes 1*, *case-names Base Step*]:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

assumes $P \Longrightarrow\hat{}\ \alpha \prec P'$
and $[\![\alpha = \tau;\ P = P']\!] \Longrightarrow Prop\ (\tau)\ P$
and $P \Longrightarrow\alpha \prec P' \Longrightarrow Prop\ \alpha\ P'$

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

lemma *weakCongTransitionWeakTransition*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

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

shows $P \Longrightarrow\hat{}\ \alpha \prec P'$
$\langle proof \rangle$

lemma *transitionWeakTransition*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

assumes $P \longmapsto\alpha \prec P'$

shows $P \Longrightarrow\hat{} \alpha \prec P'$
$\langle proof \rangle$

lemma *weakAction*:
fixes a :: *name*
and P :: *ccs*

shows $\alpha.(P) \Longrightarrow\hat{} \alpha \prec P$
$\langle proof \rangle$

lemma *weakSum1*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and Q :: *ccs*

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

shows $P \oplus Q \Longrightarrow\hat{} \alpha \prec P'$
$\langle proof \rangle$

lemma *weakSum2*:
fixes Q :: *ccs*
and α :: *act*
and Q' :: *ccs*
and P :: *ccs*

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

shows $P \oplus Q \Longrightarrow\hat{} \alpha \prec Q'$
$\langle proof \rangle$

lemma *weakPar1*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and Q :: *ccs*

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

shows $P \parallel Q \Longrightarrow\hat{} \alpha \prec P' \parallel Q$
$\langle proof \rangle$

lemma *weakPar2*:
fixes Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

and P :: *ccs*

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

shows $P \parallel Q \Longrightarrow\hat{}\ \alpha \prec P \parallel Q'$
⟨*proof*⟩

lemma *weakSync*:
fixes P :: *ccs*
and α :: *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 $P \parallel Q \Longrightarrow\hat{}\ \tau \prec P' \parallel Q'$
⟨*proof*⟩

lemma *weakRes*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*
and x :: *name*

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

shows $(\!|\nu x|\!)P \Longrightarrow\hat{}\ \alpha \prec (\!|\nu x|\!)P'$
⟨*proof*⟩

lemma *weakRepl*:
fixes P :: *ccs*
and α :: *act*
and P' :: *ccs*

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

shows $!P \Longrightarrow\alpha \prec P'$
⟨*proof*⟩

end

theory *Strong-Sim*
imports *Agent*
begin

definition *simulation* :: *ccs* ⇒ (*ccs* × *ccs*) *set* ⇒ *ccs* ⇒ *bool* (- $\rightsquigarrow$[-] - [*80*, *80*, *80*] *80*)
where
$P \rightsquigarrow[Rel]\ Q \equiv \forall\, a\ Q'.\ Q \longmapsto a \prec Q' \longrightarrow (\exists\, P'.\ P \longmapsto a \prec P' \wedge (P', Q') \in Rel)$

lemma *simI*[*case-names Sim*]:
fixes P :: *ccs*
and Rel :: (*ccs* × *ccs*) *set*
and Q :: *ccs*

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

shows $P \rightsquigarrow[Rel]\ Q$
⟨*proof*⟩

lemma *simE*:
fixes P :: *ccs*
and Rel :: (*ccs* × *ccs*) *set*
and Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

assumes $P \rightsquigarrow[Rel]\ Q$
and $Q \longmapsto\alpha \prec Q'$

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

lemma *reflexive*:
fixes P :: *ccs*
and Rel :: (*ccs* × *ccs*) *set*

assumes $Id \subseteq Rel$

shows $P \rightsquigarrow[Rel]\ P$
⟨*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 $P \rightsquigarrow[Rel]\ Q$
and $Q \rightsquigarrow[Rel']\ R$
and $Rel\ O\ Rel' \subseteq Rel''$

shows $P \leadsto[Rel''] R$
⟨*proof*⟩

end

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

definition *weakSimulation* :: $ccs \Rightarrow (ccs \times ccs)\ set \Rightarrow ccs \Rightarrow bool$ (- $\leadsto\hat{}$<-> - [*80*, *80*, *80*] *80*)
where
$P \leadsto\hat{}$<*Rel*> $Q \equiv \forall a\ Q'.\ Q \longmapsto a \prec Q' \longrightarrow (\exists P'.\ P \Longrightarrow\hat{} a \prec P' \wedge (P',\ Q') \in Rel)$

lemma *weakSimI*[*case-names Sim*]:
fixes P :: *ccs*
and *Rel* :: $(ccs \times ccs)\ set$
and Q :: *ccs*

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

shows $P \leadsto\hat{}$<*Rel*> Q
⟨*proof*⟩

lemma *weakSimE*:
fixes P :: *ccs*
and *Rel* :: $(ccs \times ccs)\ set$
and Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

assumes $P \leadsto\hat{}$<*Rel*> Q
and $Q \longmapsto\alpha \prec Q'$

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

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

assumes $Q \Longrightarrow_\tau Q'$
and $(P,\ Q) \in Rel$
and *Sim*: $\bigwedge R\ S.\ (R,\ S) \in Rel \Longrightarrow R \leadsto\hat{}$<*Rel*> S

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

$\langle proof \rangle$

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

assumes $(P,\ Q) \in Rel$
and $Q \Longrightarrow\hat{}\ \alpha \prec Q'$
and *Sim*: $\bigwedge R\ S.\ (R,\ S) \in Rel \Longrightarrow R \leadsto\hat{}<Rel>\ S$

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

lemma *reflexive*:
fixes $P \quad :: ccs$
and $Rel :: (ccs \times ccs)\ set$

assumes $Id \subseteq Rel$

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

lemma *transitive*:
fixes $P \quad :: ccs$
and $Rel \quad :: (ccs \times ccs)\ set$
and $Q \quad :: ccs$
and $Rel' \quad :: (ccs \times ccs)\ set$
and $R \quad :: ccs$
and $Rel'' :: (ccs \times ccs)\ set$

assumes $(P,\ Q) \in Rel$
and $Q \leadsto\hat{}<Rel'>\ R$
and $Rel\ O\ Rel' \subseteq Rel''$
and $\bigwedge S\ T.\ (S,\ T) \in Rel \Longrightarrow S \leadsto\hat{}<Rel>\ T$

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

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

assumes $P \leadsto\hat{}<A>\ Q$
and $A \subseteq B$

shows $P \leadsto\hat{}<B> Q$
$\langle proof \rangle$

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

assumes $P \leadsto[Rel]\ Q$

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

end

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

definition *weakCongSimulation* :: $ccs \Rightarrow (ccs \times ccs)\ set \Rightarrow ccs \Rightarrow bool$ (- $\leadsto$<-> - [*80*, *80*, *80*] *80*)
where
$P \leadsto<Rel> Q \equiv \forall a\ Q'.\ Q \longmapsto a \prec Q' \longrightarrow (\exists P'.\ P \Longrightarrow a \prec P' \wedge (P', Q') \in Rel)$

lemma *weakSimI*[*case-names Sim*]:
fixes P :: *ccs*
and Rel :: $(ccs \times ccs)$ *set*
and Q :: *ccs*

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

shows $P \leadsto<Rel> Q$
$\langle proof \rangle$

lemma *weakSimE*:
fixes P :: *ccs*
and Rel :: $(ccs \times ccs)$ *set*
and Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

assumes $P \leadsto<Rel> Q$
and $Q \longmapsto\alpha \prec Q'$

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

lemma *simWeakSim*:

fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*

assumes $P \leadsto[Rel]\ Q$

shows $P \leadsto{<}Rel{>}\ Q$
⟨*proof*⟩

lemma *weakCongSimWeakSim*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*

assumes $P \leadsto{<}Rel{>}\ Q$

shows $P \leadsto\hat{}{<}Rel{>}\ Q$
⟨*proof*⟩

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

shows $P = P' \lor (\exists P''.\ P \longmapsto \tau \prec P'' \land P'' \Longrightarrow_\tau P')$
⟨*proof*⟩

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

shows *Prop* P'
⟨*proof*⟩

lemma *simE2*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

assumes $P \leadsto{<}Rel{>}\ Q$
and $Q \Longrightarrow \alpha \prec Q'$
and *Sim*: $\bigwedge R\ S.\ (R, S) \in Rel \Longrightarrow R \leadsto\hat{}{<}Rel{>}\ S$

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

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

and *Rel* :: (*ccs* × *ccs*) *set*

assumes *Id* ⊆ *Rel*

shows $P \rightsquigarrow$<*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 $P \rightsquigarrow$<*Rel*> Q
and $Q \rightsquigarrow$<*Rel'*> R
and *Rel O Rel'* ⊆ *Rel''*
and $\bigwedge S\ T.\ (S,\ T) \in Rel \Longrightarrow S \rightsquigarrow\hat{}$<*Rel*> T

shows $P \rightsquigarrow$<*Rel''*> R
⟨*proof*⟩

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

assumes $P \rightsquigarrow$<*A*> Q
and *A* ⊆ *B*

shows $P \rightsquigarrow$<*B*> Q
⟨*proof*⟩

end

theory *Strong-Sim-SC*
imports *Strong-Sim*
begin

lemma *resNilLeft*:
fixes *x* :: *name*

shows $(\!|\nu x|\!)\mathbf{0} \rightsquigarrow[Rel]\ \mathbf{0}$
⟨*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 $x \sharp [x].P$
$\langle proof \rangle$

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

assumes $x \sharp P$
and $C1: \bigwedge y\ R.\ y \sharp R \Longrightarrow ((\!|\nu y|\!)R,\ R) \in Rel$
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 $x \sharp P$
and $C1: \bigwedge y\ R.\ y \sharp R \Longrightarrow (R,\ (\!|\nu y|\!)R) \in Rel$
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 $x \sharp P$
and $C1: \bigwedge y\ R\ T.\ y \sharp R \Longrightarrow ((\!|\nu y|\!)(R \parallel T),\ R \parallel (\!|\nu y|\!)T) \in Rel$

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 $x \sharp P$
and $C1: \bigwedge y\ R\ T.\ y \sharp R \Longrightarrow (R \parallel (\!|\nu y|\!) T,\ (\!|\nu y|\!)(R \parallel T)) \in Rel$

shows $P \parallel (\!|\nu x|\!) Q \leadsto[Rel]\ (\!|\nu x|\!)(P \parallel Q)$
⟨*proof*⟩

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

assumes $Id \subseteq Rel$

shows $P \oplus Q \leadsto[Rel]\ Q \oplus P$
⟨*proof*⟩

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

assumes $Id \subseteq Rel$

shows $(P \oplus Q) \oplus R \leadsto[Rel]\ P \oplus (Q \oplus R)$
⟨*proof*⟩

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

assumes $Id \subseteq Rel$

shows $P \oplus (Q \oplus R) \leadsto[Rel]\ (P \oplus Q) \oplus R$
⟨*proof*⟩

lemma *sumIdLeft*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs)\ set$

assumes $Id \subseteq Rel$

shows $P \oplus \mathbf{0} \leadsto[Rel]\ P$
⟨*proof*⟩

lemma *sumIdRight*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs)\ set$

assumes $Id \subseteq Rel$

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

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

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

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

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 $(P \parallel Q) \parallel R \rightsquigarrow[Rel]\ P \parallel (Q \parallel R)$
$\langle proof \rangle$

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 $P \parallel (Q \parallel R) \rightsquigarrow[Rel]\ (P \parallel Q) \parallel R$
$\langle proof \rangle$

lemma *parIdLeft*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs)\ set$

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

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

lemma *parIdRight*:
fixes $P :: ccs$
and $Rel :: (ccs \times ccs)\ set$

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 $x \sharp \alpha$
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 $x \sharp \alpha$
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*]: $!P \longmapsto \alpha \prec P'$

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 $!P \rightsquigarrow[Rel]\ P \parallel !P$
⟨*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*: $\bigwedge x\ y\ xa\ xb\ P\ Q.$
$x \leq y \Longrightarrow$
$(Q \rightsquigarrow[\{(xb,\ xa).\ x\ xb\ xa\}]\ P) \longrightarrow$
$(Q \rightsquigarrow[\{(xb,\ xa).\ y\ xb\ xa\}]\ P)$
⟨*proof*⟩

coinductive-set *bisim* :: $(ccs \times ccs)\ set$
where
$[\![P \rightsquigarrow[bisim]\ Q;\ (Q,\ P) \in bisim]\!] \Longrightarrow (P,\ Q) \in bisim$
monos *monoCoinduct*

abbreviation
bisimJudge (- $\sim$ - [*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 $\bigwedge P\ Q.\ (P,\ Q) \in X \Longrightarrow P \rightsquigarrow[(X \cup bisim)]\ Q \wedge (Q,\ P) \in X$

shows $P \sim Q$
⟨*proof*⟩

lemma *bisimCoinduct*[*consumes 1*, *case-names cSim cSym*]:
fixes P :: *ccs*
and Q :: *ccs*
and X :: (*ccs* $\times$ *ccs*) *set*

assumes $(P, Q) \in X$
and $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \leadsto[(X \cup \mathit{bisim})]\ S$
and $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow (S, R) \in X$

shows $P \sim Q$
⟨*proof*⟩

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

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

shows $P \sim Q$
⟨*proof*⟩

lemma *bisimWeakCoinduct*[*consumes 1*, *case-names cSim cSym*]:
fixes P :: *ccs*
and Q :: *ccs*
and X :: (*ccs* $\times$ *ccs*) *set*

assumes $(P, Q) \in X$
and $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow P \leadsto[X]\ Q$
and $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow (Q, P) \in X$

shows $P \sim Q$
⟨*proof*⟩

lemma *bisimE*:
fixes P :: *ccs*
and Q :: *ccs*

assumes $P \sim Q$

shows $P \leadsto[\mathit{bisim}]\ Q$
and $Q \sim P$
⟨*proof*⟩

lemma *bisimI*:
fixes P :: *ccs*
and Q :: *ccs*

assumes $P \leadsto[bisim]\ Q$
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*: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \leadsto[(bisim\ O\ X\ O\ bisim)]\ S$
and *rSym*: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow (S, R) \in X$

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*) *set*
  and a :: *name*
  and *Rel'* :: (*ccs* $\times$ *ccs*) *set*

  assumes $(P, Q) \in Rel$

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

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

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

  shows $P \oplus R \rightsquigarrow[Rel']\ Q \oplus R$
$\langle proof \rangle$

lemma *parPresAux*:
  fixes P :: *ccs*
  and Q :: *ccs*
  and *Rel* :: (*ccs* $\times$ *ccs*) *set*

  assumes $P \rightsquigarrow[Rel]\ Q$
  and $(P, Q) \in Rel$
  and $R \rightsquigarrow[Rel']\ T$
  and $(R, T) \in Rel'$
  and *C1*: $\bigwedge P'\ Q'\ R'\ T'.\ [\![(P', Q') \in Rel; (R', T') \in Rel']\!] \Longrightarrow (P' \parallel R', Q' \parallel T') \in Rel''$

  shows $P \parallel R \rightsquigarrow[Rel'']\ Q \parallel T$
$\langle proof \rangle$

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

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

shows $P \parallel R \rightsquigarrow[Rel'] \; Q \parallel R$
$\langle proof \rangle$

lemma *resPres*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*
and x :: *name*

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

shows $(\!|\nu x|\!)P \rightsquigarrow[Rel'] \; (\!|\nu x|\!)Q$
$\langle proof \rangle$

lemma *bangPres*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*

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

shows $!P \rightsquigarrow[bangRel\ Rel] \; !Q$
$\langle proof \rangle$

end

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

lemma *actPres*:
fixes P :: *ccs*
and Q :: *ccs*
and α :: *act*

assumes $P \sim Q$

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

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* $:: ccs \Rightarrow ccs \Rightarrow bool$ (- $\equiv_s$ -)
where
Refl: $P \equiv_s P$
| *Sym*: $P \equiv_s Q \Longrightarrow Q \equiv_s P$
| *Trans*: $[\![P \equiv_s Q; Q \equiv_s R]\!] \Longrightarrow P \equiv_s R$

| *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*: $x \sharp P \Longrightarrow (\!|\nu x|\!)(P \parallel Q) \equiv_s P \parallel (\!|\nu x|\!)Q$
| *ScopeExtSum*: $x \sharp P \Longrightarrow (\!|\nu x|\!)(P \oplus Q) \equiv_s P \oplus (\!|\nu x|\!)Q$
| *ScopeAct*: $x \sharp \alpha \Longrightarrow (\!|\nu x|\!)(\alpha.(P)) \equiv_s \alpha.((\!|\nu x|\!)P)$
| *ScopeCommAux*: $x \neq y \Longrightarrow (\!|\nu x|\!)((\!|\nu y|\!)P) \equiv_s (\!|\nu y|\!)((\!|\nu x|\!)P)$

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

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

  shows $(\!|\nu x|\!)((\!|\nu y|\!)P) \equiv_s (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

end

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

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

  shows $(\!|\nu x|\!)\mathbf{0} \sim \mathbf{0}$
⟨*proof*⟩

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

  assumes $x \sharp P$

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

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 $x \sharp P$

  shows $(\!|\nu x|\!)P \sim P$
$\langle proof \rangle$

lemma *scopeExtSum*:
  fixes x :: *name*
  and P :: *ccs*

and $Q :: ccs$

assumes $x \sharp P$

shows $(\!|\nu x|\!)(P \oplus Q) \sim P \oplus (\!|\nu x|\!)Q$
$\langle proof \rangle$

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

assumes $x \sharp \alpha$

shows $(\!|\nu x|\!)(\alpha.(P)) \sim \alpha.((\!|\nu x|\!)P)$
$\langle proof \rangle$

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

shows $(\!|\nu x|\!)((\!|\nu y|\!)P) \sim (\!|\nu y|\!)((\!|\nu x|\!)P)$
$\langle proof \rangle$

lemma *bangUnfold*:
fixes P

shows $!P \sim P \parallel !P$
$\langle proof \rangle$

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

assumes $P \equiv_s Q$

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

end

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

lemma *weakMonoCoinduct*: $\bigwedge x\ y\ xa\ xb\ P\ Q.$
$x \leq y \Longrightarrow$
$(Q \leadsto\hat{}<\{(xb, xa).\ x\ xb\ xa\}> P) \longrightarrow$

$(Q \leadsto\hat{}<\{(xb,\ xa).\ y\ xb\ xa\}>\ P)$
⟨*proof*⟩

coinductive-set *weakBisimulation* :: $(ccs \times ccs)\ set$
where
$[\![P \leadsto\hat{}<weakBisimulation>\ Q;\ (Q,\ P) \in weakBisimulation]\!] \Longrightarrow (P,\ Q) \in weakBisimulation$
monos *weakMonoCoinduct*

abbreviation
weakBisimJudge (- $\approx$ - [*70*, *70*] *65*) **where** $P \approx Q \equiv (P,\ Q) \in weakBisimulation$

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

assumes $(P,\ Q) \in X$
and $\bigwedge P\ Q.\ (P,\ Q) \in X \Longrightarrow P \leadsto\hat{}<(X \cup weakBisimulation)>\ Q \wedge (Q,\ P) \in X$

shows $P \approx Q$
⟨*proof*⟩

lemma *weakBisimulationCoinduct*[*consumes 1*, *case-names cSim cSym*]:
fixes $P :: ccs$
and $Q :: ccs$
and $X :: (ccs \times ccs)\ set$

assumes $(P,\ Q) \in X$
and $\bigwedge R\ S.\ (R,\ S) \in X \Longrightarrow R \leadsto\hat{}<(X \cup weakBisimulation)>\ S$
and $\bigwedge R\ S.\ (R,\ S) \in X \Longrightarrow (S,\ R) \in X$

shows $P \approx Q$
⟨*proof*⟩

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

assumes $(P,\ Q) \in X$
and $\bigwedge P\ Q.\ (P,\ Q) \in X \Longrightarrow P \leadsto\hat{}<X>\ Q \wedge (Q,\ P) \in X$

shows $P \approx Q$
⟨*proof*⟩

lemma *weakBisimWeakCoinduct*[*consumes 1*, *case-names cSim cSym*]:
fixes $P :: ccs$

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

assumes $(P, Q) \in X$
and $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow P \leadsto\hat{}{<}X{>}\ Q$
and $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow (Q, P) \in X$

shows $P \approx Q$
$\langle \mathit{proof} \rangle$

lemma *weakBisimulationE*:
fixes $P :: \mathit{ccs}$
and $Q :: \mathit{ccs}$

assumes $P \approx Q$

shows $P \leadsto\hat{}{<}\mathit{weakBisimulation}{>}\ Q$
and $Q \approx P$
$\langle \mathit{proof} \rangle$

lemma *weakBisimulationI*:
fixes $P :: \mathit{ccs}$
and $Q :: \mathit{ccs}$

assumes $P \leadsto\hat{}{<}\mathit{weakBisimulation}{>}\ Q$
and $Q \approx P$

shows $P \approx Q$
$\langle \mathit{proof} \rangle$

lemma *reflexive*:
fixes $P :: \mathit{ccs}$

shows $P \approx P$
$\langle \mathit{proof} \rangle$

lemma *symmetric*:
fixes $P :: \mathit{ccs}$
and $Q :: \mathit{ccs}$

assumes $P \approx Q$

shows $Q \approx P$
$\langle \mathit{proof} \rangle$

lemma *transitive*:
fixes $P :: \mathit{ccs}$
and $Q :: \mathit{ccs}$
and $R :: \mathit{ccs}$

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

shows $P \approx R$
⟨*proof*⟩

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

assumes $P \sim Q$

shows $P \approx Q$
⟨*proof*⟩

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

assumes $P \equiv_s Q$

shows $P \approx Q$
⟨*proof*⟩

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

assumes *PSimQ*: $P \leadsto\hat{}{<}Rel{>}\ Q$
and *QSimR*: $Q \leadsto[Rel']\ R$
and *Trans*: $Rel\ O\ Rel' \subseteq Rel''$

shows $P \leadsto\hat{}{<}Rel''{>}\ R$
⟨*proof*⟩

lemma *weakBisimWeakUpto*[*case-names cSim cSym*, *consumes 1*]:
assumes *p*: $(P, Q) \in X$
and *rSim*: $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow P \leadsto\hat{}{<}(weakBisimulation\ O\ X\ O\ bisim){>}\ Q$
and *rSym*: $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow (Q, P) \in X$

shows $P \approx Q$
⟨*proof*⟩

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

assumes p: $(P, Q) \in X$
and $rSim$: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \leadsto\hat{}<(weakBisimulation\ O\ (X \cup weakBisimulation)\ O\ bisim)> S$
and $rSym$: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow (S, R) \in X$

shows $P \approx Q$
⟨*proof*⟩

end

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

definition *weakCongruence* :: $ccs \Rightarrow ccs \Rightarrow bool$ (- $\cong$ - [70, 70] 65)
where
$P \cong Q \equiv P \leadsto<weakBisimulation> Q \wedge Q \leadsto<weakBisimulation> P$

lemma *weakCongruenceE*:
fixes P :: ccs
and Q :: ccs

assumes $P \cong Q$

shows $P \leadsto<weakBisimulation> Q$
and $Q \leadsto<weakBisimulation> P$
⟨*proof*⟩

lemma *weakCongruenceI*:
fixes P :: ccs
and Q :: ccs

assumes $P \leadsto<weakBisimulation> Q$
and $Q \leadsto<weakBisimulation> P$

shows $P \cong Q$
⟨*proof*⟩

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

assumes $Prop\ P\ Q$
and $\bigwedge P\ Q.\ Prop\ P\ Q \Longrightarrow Prop\ Q\ P$
and $\bigwedge P\ Q.\ Prop\ P\ Q \Longrightarrow (F\ P) \leadsto<weakBisimulation> (F\ Q)$

shows $F\ P \cong F\ Q$
⟨*proof*⟩

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

assumes $P \cong Q$
and $\bigwedge P\ Q.\ P \cong Q \Longrightarrow (F\ P) \leadsto$<*weakBisimulation*> $(F\ Q)$

shows $F\ P \cong F\ Q$
⟨*proof*⟩

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

shows $P \cong P$
⟨*proof*⟩

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

assumes $P \cong Q$

shows $Q \cong P$
⟨*proof*⟩

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

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

shows $P \cong R$
⟨*proof*⟩

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

assumes $P \sim Q$

shows $P \cong Q$
⟨*proof*⟩

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

assumes $P \equiv_s Q$

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

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

assumes $P \cong Q$

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

end

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

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

assumes $(P, Q) \in Rel$

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

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

assumes $P \rightsquigarrow\hat{}<Rel> Q$
and $Rel \subseteq Rel'$
and $Id \subseteq Rel'$
and $C1: \bigwedge S\ T\ U.\ (S, T) \in Rel \Longrightarrow (S \oplus U, T) \in Rel'$

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

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

and R :: *ccs*
and T :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and *Rel'* :: (*ccs* × *ccs*) *set*
and *Rel''* :: (*ccs* × *ccs*) *set*

assumes $P \rightsquigarrow\hat{}$ <*Rel*> Q
and $(P, Q) \in Rel$
and $R \rightsquigarrow\hat{}$ <*Rel'*> T
and $(R, T) \in Rel'$
and *C1*: $\bigwedge P'\ Q'\ R'\ T'.\ [\![(P', Q') \in Rel; (R', T') \in Rel']\!] \Longrightarrow (P' \parallel R', Q' \parallel T') \in Rel''$

shows $P \parallel R \rightsquigarrow\hat{}$ <*Rel''*> $Q \parallel T$
⟨*proof*⟩

lemma *parPres*:
fixes P :: *ccs*
and Q :: *ccs*
and R :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and *Rel'* :: (*ccs* × *ccs*) *set*
assumes $P \rightsquigarrow\hat{}$ <*Rel*> Q
and $(P, Q) \in Rel$
and *C1*: $\bigwedge S\ T\ U.\ (S, T) \in Rel \Longrightarrow (S \parallel U, T \parallel U) \in Rel'$

shows $P \parallel R \rightsquigarrow\hat{}$ <*Rel'*> $Q \parallel R$
⟨*proof*⟩

lemma *resPres*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*
and x :: *name*

assumes $P \rightsquigarrow\hat{}$ <*Rel*> Q
and $\bigwedge R\ S\ y.\ (R, S) \in Rel \Longrightarrow ((\!|\nu y|\!)R, (\!|\nu y|\!)S) \in Rel'$

shows $(\!|\nu x|\!)P \rightsquigarrow\hat{}$ <*Rel'*> $(\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *bangPres*:
fixes P :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and Q :: *ccs*

assumes $(P, Q) \in Rel$
and *C1*: $\bigwedge R\ S.\ (R, S) \in Rel \Longrightarrow R \rightsquigarrow\hat{}$ <*Rel*> S
and *Par*: $\bigwedge R\ S\ T\ U.\ [\![(R, S) \in Rel; (T, U) \in Rel']\!] \Longrightarrow (R \parallel T, S \parallel U) \in$

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

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

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 $\alpha.(P) \approx \alpha.(Q)$
$\langle proof \rangle$

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

assumes $P \approx Q$

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

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

assumes $P \approx Q$

shows $(\!|\nu x|\!)P \approx (\!|\nu x|\!)Q$
$\langle proof \rangle$

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 :: $(ccs \times ccs)$ *set*
and a :: *name*
and Rel' :: $(ccs \times ccs)$ *set*

assumes $(P, Q) \in Rel$

shows $\alpha.(P) \rightsquigarrow<Rel> \alpha.(Q)$
⟨*proof*⟩

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

assumes $P \rightsquigarrow<Rel> Q$
and $(P, Q) \in Rel$
and $C1$: $\bigwedge S\ T\ U.\ (S, T) \in Rel \Longrightarrow (S \parallel U, T \parallel U) \in Rel'$

shows $P \parallel R \rightsquigarrow<Rel'> Q \parallel R$
⟨*proof*⟩

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

and x :: *name*

assumes $P \rightsquigarrow$<*Rel*> Q
and $\bigwedge R\ S\ y.\ (R,\ S) \in Rel \Longrightarrow ((\!|\nu y|\!)R,\ (\!|\nu y|\!)S) \in Rel'$

shows $(\!|\nu x|\!)P \rightsquigarrow$<*Rel'*> $(\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *bangPres*:
fixes P :: *ccs*
and Q :: *ccs*
and *Rel* :: (*ccs* × *ccs*) *set*
and *Rel'* :: (*ccs* × *ccs*) *set*

assumes $(P,\ Q) \in Rel$
and *C1*: $\bigwedge R\ S.\ (R,\ S) \in Rel \Longrightarrow R \rightsquigarrow$<*Rel'*> S
and *C2*: $Rel \subseteq Rel'$

shows $!P \rightsquigarrow$<*bangRel Rel'*> $!Q$
⟨*proof*⟩

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 α :: *act*

assumes $P \cong Q$

shows $\alpha.(P) \cong \alpha.(Q)$
⟨*proof*⟩

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

assumes $P \cong Q$

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

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.