The Calculus of Communicating Systems

Jesper Bengtson

May 26, 2024

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* (‹-›) *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** *coAction* $\alpha \neq \tau = (\alpha \neq \tau)$ **and** (*coAction* $\alpha = \tau) = (\alpha = \tau)$
⟨*proof*⟩

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

shows $x \sharp$ *coAction* $a = x \sharp a$
⟨*proof*⟩

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

assumes $y \sharp P$

shows $(\!|\nu x|\!)P = (\!|\nu y|\!)([(x, y)] \cdot P)$
⟨*proof*⟩

inductive *semantics* :: *ccs* ⇒ *act* ⇒ *ccs* ⇒ *bool* (- ⟼- ≺ - [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*
⟨*proof*⟩

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

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

$\Longrightarrow Prop\ (\mathcal{C}::'a::fs\text{-}name)\ R\ \beta\ R'$
⟨*proof*⟩

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

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** $x \sharp P'$
⟨*proof*⟩

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 $Prop\ \alpha\ P$

shows $Prop\ \beta\ P'$
⟨*proof*⟩

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 $\bigwedge P'.\ P \longmapsto \alpha \prec P' \Longrightarrow Prop\ P'$
and $\bigwedge Q'.\ Q \longmapsto \alpha \prec Q' \Longrightarrow Prop\ Q'$

shows $Prop\ R$
⟨*proof*⟩

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

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

shows $Prop\ \alpha\ R$
⟨*proof*⟩

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

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

shows $Prop\ P'$
⟨*proof*⟩

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 $\mathcal{C} :: 'a::fs\text{-}name$

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

$\| !P)\ \alpha\ (P \parallel P')$
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 $P \Longrightarrow_\tau P' \equiv (P, P') \in \{(P, P') \mid P\ P'.\ P \longmapsto \tau \prec P'\}^*$

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* (- ⟹- ≺ - [*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 $P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

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

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

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 α :: *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 α :: *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 α :: *act*

and $P' :: ccs$
and $Q :: ccs$

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

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

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

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

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

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

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

shows $P \parallel Q \Longrightarrow \tau \prec P' \parallel Q'$
⟨*proof*⟩

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

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

shows $(\!|\nu x|\!)P \Longrightarrow \alpha \prec (\!|\nu x|\!)P'$
⟨*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* ⇒ *act* ⇒ *ccs* ⇒ *bool* (- $\Longrightarrow\hat{}$ - ≺ - [*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 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* α P'
⟨*proof*⟩

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

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

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

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

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

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

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

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

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

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

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

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

end

theory *Strong-Sim*
imports *Agent*
begin

definition *simulation* :: $ccs \Rightarrow (ccs \times ccs)\ set \Rightarrow ccs \Rightarrow bool$ (- $\rightsquigarrow$[-] - [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 :: $(ccs \times ccs)\ set$
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 :: $(ccs \times ccs)\ set$
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 :: $(ccs \times ccs)\ set$

assumes $Id \subseteq Rel$

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

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

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

shows $P \rightsquigarrow[Rel'^{\frown}] R$
⟨*proof*⟩

end

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

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

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

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

shows $P \rightsquigarrow^{\frown} <Rel> Q$
⟨*proof*⟩

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

assumes $P \rightsquigarrow^{\frown} <Rel> Q$
and $Q \longmapsto \alpha \prec Q'$

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

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

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

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

⟨*proof*⟩

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

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

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

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

assumes $Id \subseteq Rel$

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

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

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

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

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

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

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

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

assumes $P \rightsquigarrow[Rel] Q$

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

end

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

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

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 $P \rightsquigarrow<Rel> Q$
⟨*proof*⟩

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

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

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

lemma *simWeakSim*:

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

assumes $P \rightsquigarrow[Rel]\ Q$

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

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

assumes $P \rightsquigarrow<Rel>\ Q$

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

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

shows $P = P' \vee (\exists\, P''.\ P \longmapsto \tau \prec P'' \wedge P'' \Longrightarrow_\tau P')$
⟨*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 :: $(ccs \times ccs)$ *set*
and Q :: *ccs*
and α :: *act*
and Q' :: *ccs*

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

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

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

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

assumes $Id \subseteq Rel$

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

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

assumes $P \rightsquigarrow <Rel> Q$
and $Q \rightsquigarrow <Rel'> R$
and $Rel\ O\ Rel' \subseteq Rel''$
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 $A :: (ccs \times ccs)\ set$
and $Q :: ccs$
and $B :: (ccs \times ccs)\ set$

assumes $P \rightsquigarrow <A> Q$
and $A \subseteq B$

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

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 $\mathbf{0} \rightsquigarrow[Rel]\ (\!|\nu x|\!)\mathbf{0}$
⟨*proof*⟩

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

shows $x \sharp [x].P$
⟨*proof*⟩

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 $(\!|\nu x|\!)(P \oplus Q) \rightsquigarrow[Rel]\ P \oplus (\!|\nu x|\!)\,Q$
⟨*proof*⟩

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 $P \oplus (\!|\nu x|\!)\,Q \rightsquigarrow[Rel]\ (\!|\nu x|\!)(P \oplus Q)$
⟨*proof*⟩

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

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

and $Q :: ccs$

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

shows $P \parallel (\!|\nu x|\!)\, Q \rightsquigarrow [Rel]\ (\!|\nu x|\!)(P \parallel Q)$
⟨*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 $Rel :: (ccs \times ccs)\ set$

assumes $Id \subseteq Rel$

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

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

assumes $Id \subseteq Rel$

shows $P \rightsquigarrow [Rel]\ P \oplus \mathbf{0}$
⟨*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 $Rel :: (ccs \times ccs)\ set$

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 $Rel :: (ccs \times ccs)\ set$

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

shows $P \rightsquigarrow[Rel]\ P \parallel \mathbf{0}$
⟨*proof*⟩

declare *fresh-atm*[*simp*]

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

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

shows $(\!|\nu x|\!)(\alpha.(P)) \rightsquigarrow[Rel]\ (\alpha.((\!|\nu x|\!)P))$
⟨*proof*⟩

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

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

shows $\alpha.((\!|\nu x|\!)P) \rightsquigarrow[Rel]\ (\!|\nu x|\!)(\alpha.(P))$
⟨*proof*⟩

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 $(\!|\nu x|\!)((\!|\nu y|\!)P) \rightsquigarrow[Rel]\ (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

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

lemma *bangUnfoldLeft*:
fixes $P :: ccs$

assumes $Id \subseteq Rel$

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

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 $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \rightsquigarrow [(X \cup bisim)]\ S$
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 $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \rightsquigarrow [X]\ S \wedge (S, R) \in X$

shows $P \sim Q$
⟨*proof*⟩

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

assumes $(P, Q) \in X$
and $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow P \rightsquigarrow [X]\ Q$
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 $P \rightsquigarrow [bisim]\ Q$
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*: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \rightsquigarrow[(bisim\ O\ X\ O\ bisim)]\ S$
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 $\alpha.(P) \rightsquigarrow[Rel] \alpha.(Q)$
⟨*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 *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 $P \parallel R \rightsquigarrow[Rel''] Q \parallel T$
⟨*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 Rel :: $(ccs \times ccs)$ *set*
and Q :: *ccs*

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

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

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 $\alpha.(P) \sim \alpha.(Q)$
⟨*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* ⇒ *ccs* ⇒ *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 $P \oplus Q \sim Q \oplus P$

⟨*proof*⟩

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

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

lemma *sumId*:
fixes P :: *ccs*

shows $P \oplus \mathbf{0} \sim P$
⟨*proof*⟩

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

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

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

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

lemma *parId*:
fixes P :: *ccs*

shows $P \parallel \mathbf{0} \sim P$
⟨*proof*⟩

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

assumes $x \sharp P$

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

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

and $Q :: ccs$

assumes $x \sharp P$

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

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

assumes $x \sharp \alpha$

shows $(\!|\nu x|\!)(\alpha.(P)) \sim \alpha.((\!|\nu x|\!)P)$
⟨*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 $!P \sim P \parallel !P$
⟨*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*: $\bigwedge x\ y\ xa\ xb\ P\ Q.$
$x \leq y \Longrightarrow$
$(Q \rightsquigarrow\hat{} <\{(xb,\ xa).\ x\ xb\ xa\}>\ P) \longrightarrow$

$$(Q \leadsto\hat{}\!<\{(xb,\ xa).\ y\ xb\ xa\}>\ P)$$

⟨*proof*⟩

coinductive-set *weakBisimulation* :: (*ccs* × *ccs*) *set*
where
⟦$P \leadsto\hat{}\!<$*weakBisimulation*$>\ Q$; $(Q,\ P) \in$ *weakBisimulation*⟧ ⟹ $(P,\ Q) \in$ *weakBisimulation*
monos *weakMonoCoinduct*

abbreviation
weakBisimJudge (- ≈ - [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* × *ccs*) *set*

assumes $(P,\ Q) \in X$
and ⋀$P\ Q.\ (P,\ Q) \in X \Longrightarrow P \leadsto\hat{}\!<(X \cup$ *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* × *ccs*) *set*

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

shows $P \approx Q$
⟨*proof*⟩

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

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

shows $P \approx Q$
⟨*proof*⟩

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

and $Q :: ccs$
and $X :: (ccs \times ccs)\ set$

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

shows $P \approx Q$
⟨*proof*⟩

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

assumes $P \approx Q$

shows $P \rightsquigarrow\hat{}<weakBisimulation> Q$
and $Q \approx P$
⟨*proof*⟩

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

assumes $P \rightsquigarrow\hat{}<weakBisimulation> Q$
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 $P \approx R$
⟨*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*: $P \rightsquigarrow\hat{}<Rel> Q$
and *QSimR*: $Q \rightsquigarrow[Rel'] R$
and *Trans*: $Rel\ O\ Rel' \subseteq Rel''$

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

lemma *weakBisimWeakUpto*[*case-names cSim cSym*, *consumes 1*]:
assumes *p*: $(P, Q) \in X$
and *rSim*: $\bigwedge P\ Q.\ (P, Q) \in X \Longrightarrow P \rightsquigarrow\hat{}<(weakBisimulation\ O\ X\ O\ bisim)> Q$
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*: $\bigwedge R\ S.\ (R, S) \in X \Longrightarrow R \rightsquigarrow\hat{}<(weakBisimulation\ O\ (X \cup weakBisimulation)\ O\ bisim)> S$
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* ⇒ *ccs* ⇒ *bool* (- ≅ - [70, 70] 65)
where
$P \cong Q \equiv P \rightsquigarrow<weakBisimulation> Q \wedge Q \rightsquigarrow<weakBisimulation> P$

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

assumes $P \cong Q$

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

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

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

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 $\bigwedge P\ Q.\ Prop\ P\ Q \Longrightarrow (F\ P) \rightsquigarrow<weakBisimulation> (F\ Q)$

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 $\bigwedge P\ Q.\ P \cong Q \Longrightarrow (F\ P) \rightsquigarrow <weakBisimulation> (F\ Q)$

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 $P \cong Q$
⟨*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* :: $(ccs \times ccs)$ *set*
and a :: *name*
and *Rel'* :: $(ccs \times ccs)$ *set*

assumes $(P, Q) \in Rel$

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

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

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

and $R :: ccs$
and $T :: ccs$
and $Rel :: (ccs \times ccs)\ set$
and $Rel' :: (ccs \times ccs)\ set$
and $Rel'' :: (ccs \times ccs)\ set$

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 $Rel :: (ccs \times ccs)\ set$
and $Rel' :: (ccs \times ccs)\ set$
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 $Rel :: (ccs \times ccs)\ set$
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 $Rel :: (ccs \times ccs)\ set$
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 $!P \leadsto^{\hat{}} <Rel'> !Q$
⟨*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 α :: *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 :: $(ccs \times ccs)$ *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 :: $(ccs \times ccs)\ set$
and Rel' :: $(ccs \times ccs)\ set$

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 $P \parallel R \cong Q \parallel R$
⟨*proof*⟩

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

assumes $P \cong Q$

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

lemma *weakBisimBangRel*: *bangRel weakBisimulation* $\subseteq$ *weakBisimulation*
⟨*proof*⟩

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

assumes $P \cong Q$

shows $!P \cong !Q$
⟨*proof*⟩

end

References

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