The pi-calculus

Jesper Bengtson

March 17, 2025

Abstract

Contents

1 Overview

These theories formalise the following results for psi-calculi. Note that there is only an early semantics for psi-calculi, although a late one may appear later.

- strong bisimilarity is preserved by all operators except the input-prefix
- strong equivalence is a congruence
- weak bisimilarity is preserved by all operators except case and the input-prefix
- weak congruence is a congruence
- strong equivalence respect the laws of structural congruence
- all strongly equivalent agents are also weakly congruent which in turn are weakly bisimilar. Moreover, strongly equivalent agents are also strongly bisimilar
- as a corollary of the last two points, all mentioned equivalences respect the law of structural congruence
- for instances of psi-calculi where assertion composition satisfies weakening, the definition of weak bisimilarity can be simplified significantly and proven equivalent to the version that applies when weakening does not hold

- for certain versions of psi-calculi, sum can be encoded
- for certain versions of psi-calculi, the tau-prefix can be encoded and when weakening is satisfied, all of the tau-laws hold.

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; files with the prefix *Weaken* cover theories where weakening holds for the static implication; 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 simulation or bisimulation being preserved by a certain operator; files with the suffix *StructCong* reason about structural congruence.

For a complete exposition of all of theories, please consult Bengtson's Ph. D. thesis [1]. A shorter presentation can be found in our TPHOLs paper 'Psi-calculi in Isabelle' from 2009 [3]. There are also two LICS-papers that focus on the mathematical theories, rather than the Isabelle formalisations [2, 4].

2 Formalisation

theory *Chain*
imports *HOL−Nominal.Nominal*
begin

lemma *pt-set-nil*:
fixes *Xs* :: *'a set*
assumes *pt*: *pt TYPE('a) TYPE ('x)*
and *at*: *at TYPE('x)*

shows ([]::*'x prm*)•*Xs* = *Xs*
⟨*proof*⟩

lemma *pt-set-append*:
fixes *pi1* :: *'x prm*
and *pi2* :: *'x prm*
and *Xs* :: *'a set*
assumes *pt*: *pt TYPE('a) TYPE ('x)*
and *at*: *at TYPE('x)*

shows (*pi1*@*pi2*)•*Xs* = *pi1*•(*pi2*•*Xs*)
⟨*proof*⟩

lemma *pt-set-prm-eq*:

fixes *pi1* :: *'x prm*
and *pi2* :: *'x prm*
and *Xs* :: *'a set*
assumes *pt*: *pt TYPE('a) TYPE ('x)*
and *at*: *at TYPE('x)*

shows *pi1* $\triangleq$ *pi2* $\Longrightarrow$ *pi1*$\cdot$*Xs* = *pi2*$\cdot$*Xs*
⟨*proof*⟩

lemma *pt-set-inst*:
assumes *pt*: *pt TYPE('a) TYPE ('x)*
and *at*: *at TYPE('x)*

shows *pt TYPE('a set) TYPE('x)*
⟨*proof*⟩

lemma *pt-ball-eqvt*:
fixes *pi* :: *'a prm*
and *Xs* :: *'b set*
and *P* :: *'b* ⇒ *bool*

assumes *pt*: *pt TYPE('b) TYPE('a)*
and *at*: *at TYPE('a)*

shows (*pi* ∙ (∀ *x* ∈ *Xs*. *P x*)) = (∀ *x* ∈ (*pi* ∙ *Xs*). *pi* ∙ *P* (*rev pi* ∙ *x*))
⟨*proof*⟩

lemma *perm-cart-prod*:
fixes *Xs* :: *'b set*
and *Ys* :: *'c set*
and *p* :: *'a prm*

assumes *pt1*: *pt TYPE('b) TYPE('a)*
and *pt2*: *pt TYPE('c) TYPE('a)*
and *at*: *at TYPE('a)*

shows (*p* ∙ (*Xs* × *Ys*)) = (((*p* ∙ *Xs*) × (*p* ∙ *Ys*))::(('*b* × *'c*) *set*))
⟨*proof*⟩

lemma *supp-member*:
fixes *Xs* :: *'b set*
and *x* :: *'a*

assumes *pt*: *pt TYPE('b) TYPE('a)*
and *at*: *at TYPE('a)*
and *fs*: *fs TYPE('b) TYPE('a)*
and *finite Xs*
and *x* ∈ ((*supp Xs*)::*'a set*)

obtains X **where** $(X::'b) \in Xs$ **and** $x \in supp\ X$
⟨*proof*⟩

lemma *supp-cart-prod-empty*[*simp*]:
fixes $Xs :: 'b\ set$

shows $supp\ (Xs \times \{\}) = (\{\}::'a\ set)$
and $supp\ (\{\} \times Xs) = (\{\}::'a\ set)$
⟨*proof*⟩

lemma *supp-cart-prod*:
fixes $Xs :: 'b\ set$
and $Ys :: 'c\ set$

assumes *pt1*: $pt\ TYPE('b)\ TYPE('a)$
and *pt2*: $pt\ TYPE('c)\ TYPE('a)$
and *fs1*: $fs\ TYPE('b)\ TYPE('a)$
and *fs2*: $fs\ TYPE('c)\ TYPE('a)$
and *at*: $at\ TYPE('a)$
and *f1*: $finite\ Xs$
and *f2*: $finite\ Ys$
and *a*: $Xs \neq \{\}$
and *b*: $Ys \neq \{\}$

shows $((supp\ (Xs \times Ys))::'a\ set) = ((supp\ Xs) \cup (supp\ Ys))$
⟨*proof*⟩

lemma *fresh-cart-prod*:
fixes $x :: 'a$
and $Xs :: 'b\ set$
and $Ys :: 'c\ set$

assumes *pt1*: $pt\ TYPE('b)\ TYPE('a)$
and *pt2*: $pt\ TYPE('c)\ TYPE('a)$
and *fs1*: $fs\ TYPE('b)\ TYPE('a)$
and *fs2*: $fs\ TYPE('c)\ TYPE('a)$
and *at*: $at\ TYPE('a)$
and *f1*: $finite\ Xs$
and *f2*: $finite\ Ys$
and *a*: $Xs \neq \{\}$
and *b*: $Ys \neq \{\}$

shows $(x \sharp (Xs \times Ys)) = (x \sharp Xs \wedge x \sharp Ys)$
⟨*proof*⟩

lemma *fresh-star-cart-prod*:
fixes $Zs :: 'a\ set$
and $xvec :: 'a\ list$
and $Xs :: 'b\ set$

and $Ys :: 'c\ set$

assumes *pt1*: $pt\ TYPE('b)\ TYPE('a)$
and *pt2*: $pt\ TYPE('c)\ TYPE('a)$
and *fs1*: $fs\ TYPE('b)\ TYPE('a)$
and *fs2*: $fs\ TYPE('c)\ TYPE('a)$
and *at*: $at\ TYPE('a)$
and *f1*: $finite\ Xs$
and *f2*: $finite\ Ys$
and *a*: $Xs \neq \{\}$
and *b*: $Ys \neq \{\}$

shows $(Zs\ \sharp*\ (Xs \times Ys)) = (Zs\ \sharp*\ Xs \wedge Zs\ \sharp*\ Ys)$
and $(xvec\ \sharp*\ (Xs \times Ys)) = (xvec\ \sharp*\ Xs \wedge xvec\ \sharp*\ Ys)$
⟨*proof*⟩

lemma *permCommute*:
fixes $p :: 'a\ prm$
and $q :: 'a\ prm$
and $P :: 'x$
and $Xs :: 'a\ set$
and $Ys :: 'a\ set$

assumes *pt*: $pt\ TYPE('x)\ TYPE('a)$
and *at*: $at\ TYPE('a)$
and *a*: $(set\ p) \subseteq Xs \times Ys$
and *b*: $Xs\ \sharp*\ q$
and *c*: $Ys\ \sharp*\ q$

shows $p \cdot q \cdot P = q \cdot p \cdot P$
⟨*proof*⟩

definition
$distinctPerm :: 'a\ prm \Rightarrow bool$ **where**
$distinctPerm\ p \equiv distinct((map\ fst\ p)@(map\ snd\ p))$

lemma *at-set-avoiding-aux'*:
fixes $Xs::'a\ set$
and $As::'a\ set$
assumes *at*: $at\ TYPE('a)$
and *a*: $finite\ Xs$
and *b*: $Xs \subseteq As$
and *c*: $finite\ As$
and *d*: $finite\ ((supp\ c)::'a\ set)$
shows $\exists(Ys::'a\ set)\ (pi::'a\ prm).\ Ys\sharp*c \wedge Ys \cap As = \{\} \wedge (pi \cdot Xs = Ys) \wedge$
$set\ pi \subseteq Xs \times Ys \wedge finite\ Ys \wedge (distinctPerm\ pi)$
⟨*proof*⟩

lemma *at-set-avoiding*:
fixes $Xs::'a\ set$
assumes *at*: $at\ TYPE('a)$
and *a*: $finite\ Xs$
and *b*: $finite\ ((supp\ c)::'a\ set)$
obtains $pi::'a\ prm$ **where** $(pi \cdot Xs)\ \sharp*\ c$ **and** $set\ pi \subseteq Xs \times (pi \cdot Xs)$ **and** $distinctPerm\ pi$
⟨*proof*⟩

lemma *pt-swap*:
fixes $x :: 'a$
and $a :: 'x$
and $b :: 'x$

assumes *pt*: $pt\ TYPE('a)\ TYPE('x)$
and *at*: $at\ TYPE('x)$

shows $[(a,\ b)] \cdot x = [(b,\ a)] \cdot x$
⟨*proof*⟩

atom-decl *name*

lemma *supp-subset*:
fixes $Xs :: 'a::fs\text{-}name\ set$
and $Ys :: 'a::fs\text{-}name\ set$

assumes $Xs \subseteq Ys$
and $finite\ Xs$
and $finite\ Ys$

shows $(supp\ Xs) \subseteq ((supp\ Ys)::name\ set)$
⟨*proof*⟩

abbreviation *mem-def* $:: 'a \Rightarrow 'a\ list \Rightarrow bool\ (\langle\text{-}\ mem\ \text{-}\rangle\ [80,\ 80]\ 80)$ **where**
$x\ mem\ xs \equiv x \in set\ xs$

lemma *memFresh*:
fixes $x :: name$
and $p :: 'a::fs\text{-}name$
and $l :: ('a::fs\text{-}name)\ list$

assumes $x \sharp l$
and $p\ mem\ l$

shows $x \sharp p$
⟨*proof*⟩

lemma *memFreshChain*:

fixes *xvec* :: *name list*
and *p* :: *'a::fs-name*
and *l* :: *'a::fs-name list*
and *Xs* :: *name set*

assumes *p mem l*

shows *xvec* $\sharp*$ *l* $\Longrightarrow$ *xvec* $\sharp*$ *p*
and *Xs* $\sharp*$ *l* $\Longrightarrow$ *Xs* $\sharp*$ *p*
⟨*proof*⟩

lemma *fresh-star-list-append*[*simp*]:
fixes *A* :: *name list*
and *B* :: *name list*
and *C* :: *name list*

shows (*A* $\sharp*$ (*B* @ *C*)) = ((*A* $\sharp*$ *B*) $\wedge$ (*A* $\sharp*$ *C*))
⟨*proof*⟩

lemma *unionSimps*[*simp*]:
fixes *Xs* :: *name set*
and *Ys* :: *name set*
and *C* :: *'a::fs-name*

shows ((*Xs* $\cup$ *Ys*) $\sharp*$ *C*) = ((*Xs* $\sharp*$ *C*) $\wedge$ (*Ys* $\sharp*$ *C*))
⟨*proof*⟩

lemma *substFreshAux*[*simp*]:
fixes *C* :: *'a::fs-name*
and *xvec* :: *name list*

shows *xvec* $\sharp*$ (*supp C* − *set xvec*)
⟨*proof*⟩

lemma *fresh-star-perm-app*[*simp*]:
fixes *Xs* :: *name set*
and *xvec* :: *name list*
and *p* :: *name prm*
and *C* :: *'d::fs-name*

shows ⟦*Xs* $\sharp*$ *p*; *Xs* $\sharp*$ *C*⟧ $\Longrightarrow$ *Xs* $\sharp*$ (*p* $\cdot$ *C*)
and ⟦*xvec* $\sharp*$ *p*; *xvec* $\sharp*$ *C*⟧ $\Longrightarrow$ *xvec* $\sharp*$ (*p* $\cdot$ *C*)
⟨*proof*⟩

lemma *freshSets*[*simp*]:
fixes *x* :: *name*
and *y* :: *name*
and *xvec* :: *name list*
and *X* :: *name set*

and C :: $'a$

shows ([]::*name list*) $\sharp*$ C
and ([]::*name list*) $\sharp*$ $[y].C$
and ({}::*name set*) $\sharp*$ C
and ({}::*name set*) $\sharp*$ $[y].C$
and $((x\#xvec)\ \sharp*\ C) = (x\ \sharp\ C \wedge xvec\ \sharp*\ C)$
and $((x\#xvec)\ \sharp*\ ([y].C)) = (x\ \sharp\ ([y].C) \wedge xvec\ \sharp*\ ([y].C))$
and $((insert\ x\ X)\ \sharp*\ C) = (x\ \sharp\ C \wedge X\ \sharp*\ C)$
and $((insert\ x\ X)\ \sharp*\ ([y].C)) = (x\ \sharp\ ([y].C) \wedge X\ \sharp*\ ([y].C))$
⟨*proof*⟩

lemma *freshStarAtom*[*simp*]: (*xvec*::*name list*) $\sharp*$ (*x*::*name*) = $x\ \sharp\ xvec$
⟨*proof*⟩

lemma *name-list-set-fresh*[*simp*]:
fixes *xvec* :: *name list*
and x :: $'a$::*fs-name*

shows (*set xvec*) $\sharp*$ x = $xvec\ \sharp*\ x$
⟨*proof*⟩

lemma *name-list-supp*:
fixes *xvec* :: *name list*

shows *set xvec* = *supp xvec*
⟨*proof*⟩

lemma *abs-fresh-list-star*:
fixes *xvec* :: *name list*
and a :: *name*
and P :: $'a$::*fs-name*

shows ($xvec\ \sharp*\ [a].P$) = ((*set xvec*) − {a}) $\sharp*$ P
⟨*proof*⟩

lemma *abs-fresh-set-star*:
fixes X :: *name set*
and a :: *name*
and P :: $'a$::*fs-name*

shows ($X\ \sharp*\ [a].P$) = (X − {a}) $\sharp*$ P
⟨*proof*⟩

lemmas *abs-fresh-star* = *abs-fresh-list-star abs-fresh-set-star*

lemma *abs-fresh-list-star'*[*simp*]:
fixes *xvec* :: *name list*
and a :: *name*

and P :: *'a::fs-name*

assumes $a \sharp xvec$

shows $xvec \sharp* [a].P = xvec \sharp* P$
⟨*proof*⟩

lemma *freshChainSym*[*simp*]:
fixes *xvec* :: *name list*
and *yvec* :: *name list*

shows $xvec \sharp* yvec = yvec \sharp* xvec$
⟨*proof*⟩

lemmas [*eqvt*] = *perm-cart-prod*[*OF pt-name-inst*, *OF pt-name-inst*, *OF at-name-inst*]

lemma *name-set-avoiding*:
fixes *c* :: *'a::fs-name*
and *X* :: *name set*

assumes *finite X*
and $\bigwedge pi::name\ prm.\ [\![(pi \cdot X) \sharp* c;\ distinctPerm\ pi;\ set\ pi \subseteq X \times (pi \cdot X)]\!]$
$\Longrightarrow$ *thesis*

shows *thesis*
⟨*proof*⟩

lemmas *simps*[*simp*] = *fresh-atm fresh-prod*
pt3[*OF pt-name-inst*, *OF at-ds1*, *OF at-name-inst*]
pt-fresh-fresh[*OF pt-name-inst*, *OF at-name-inst*]
pt-rev-pi[*OF pt-name-inst*, *OF at-name-inst*]
pt-pi-rev[*OF pt-name-inst*, *OF at-name-inst*]

lemmas *name-supp-cart-prod* = *supp-cart-prod*[*OF pt-name-inst*, *OF pt-name-inst*,
OF fs-name-inst, *OF fs-name-inst*, *OF at-name-inst*]
lemmas *name-fresh-cart-prod* = *fresh-cart-prod*[*OF pt-name-inst*, *OF pt-name-inst*,
OF fs-name-inst, *OF fs-name-inst*, *OF at-name-inst*]
lemmas *name-fresh-star-cart-prod* = *fresh-star-cart-prod*[*OF pt-name-inst*, *OF pt-name-inst*,
OF fs-name-inst, *OF fs-name-inst*, *OF at-name-inst*]

lemmas *name-swap-bij*[*simp*] = *pt-swap-bij*[*OF pt-name-inst*, *OF at-name-inst*]
lemmas *name-swap* = *pt-swap*[*OF pt-name-inst*, *OF at-name-inst*]
lemmas *name-set-fresh-fresh*[*simp*] = *pt-freshs-freshs*[*OF pt-name-inst*, *OF at-name-inst*]
lemmas *list-fresh*[*simp*] = *fresh-list-nil fresh-list-cons fresh-list-append*

definition *eqvt* :: *'a::fs-name set* ⇒ *bool* **where**
eqvt X ≡ ∀ *x* ∈ *X*. ∀ *p::name prm*. *p* · *x* ∈ *X*

lemma *eqvtUnion*[*intro*]:
fixes *Rel* :: ('*d*::*fs-name*) *set*
and *Rel'* :: '*d set*

assumes *EqvtRel*: *eqvt Rel*
and *EqvtRel'*: *eqvt Rel'*

shows *eqvt* (*Rel* ∪ *Rel'*)
⟨*proof*⟩

lemma *eqvtPerm*[*simp*]:
fixes *X* :: ('*d*::*fs-name*) *set*
and *x* :: *name*
and *y* :: *name*

assumes *eqvt X*

shows ([(*x*, *y*)] ∙ *X*) = *X*
⟨*proof*⟩

lemma *eqvtI*:
fixes *X* :: '*d*::*fs-name set*
and *x* :: '*d*
and *p* :: *name prm*

assumes *eqvt X*
and *x* ∈ *X*

shows (*p* ∙ *x*) ∈ *X*
⟨*proof*⟩

lemma *fresh-star-list-nil*[*simp*]:
fixes *xvec* :: *name list*
and *Xs* :: *name set*

shows *xvec* ♯* []
and *Xs* ♯* []
⟨*proof*⟩

lemma *fresh-star-list-cons*[*simp*]:
fixes *xvec* :: *name list*
and *Xs* :: *name set*
and *x* :: '*a*::*fs-name*
and *xs* :: '*a list*

shows (*xvec* ♯* (*x*#*xs*)) = ((*xvec* ♯* *x*) ∧ *xvec* ♯* *xs*)
and (*Xs* ♯* (*x*#*xs*)) = ((*Xs* ♯* *x*) ∧ (*Xs* ♯* *xs*))
⟨*proof*⟩

lemma *freshStarPair*[*simp*]:
fixes X :: *name set*
and *xvec* :: *name list*
and x :: *'a::fs-name*
and y :: *'b::fs-name*

shows $(X \sharp* (x, y)) = (X \sharp* x \wedge X \sharp* y)$
and $(xvec \sharp* (x, y)) = (xvec \sharp* x \wedge xvec \sharp* y)$
⟨*proof*⟩
lemma *name-list-avoiding*:
fixes c :: *'a::fs-name*
and *xvec* :: *name list*

assumes $\bigwedge pi::name\ prm.\ [\![(pi \cdot xvec) \sharp* c; distinctPerm\ pi; set\ pi \subseteq (set\ xvec) \times (set\ (pi \cdot xvec))]\!] \Longrightarrow thesis$

shows *thesis*
⟨*proof*⟩

lemma *distinctPermSimps*[*simp*]:
fixes p :: *name prm*
and a :: *name*
and b :: *name*

shows *distinctPerm*([]::*name prm*)
and $(distinctPerm((a, b)\#p)) = (distinctPerm\ p \wedge a \neq b \wedge a \sharp p \wedge b \sharp p)$
⟨*proof*⟩

lemma *map-eqvt*[*eqvt*]:
fixes p :: *name prm*
and *lst* :: *'a::pt-name list*

shows $(p \cdot (map\ f\ lst)) = map\ (p \cdot f)\ (p \cdot lst)$
⟨*proof*⟩

lemma *consPerm*:
fixes x :: *name*
and y :: *name*
and p :: *name prm*
and C :: *'a::pt-name*

shows $((x, y)\#p) \cdot C = [(x, y)] \cdot p \cdot C$
⟨*proof*⟩

⟨*ML*⟩

lemma *distinctEqvt*[*eqvt*]:
fixes p :: *name prm*
and *xs* :: *'a::pt-name list*

shows $(p \cdot (distinct\ xs)) = distinct\ (p \cdot xs)$
⟨*proof*⟩

lemma *distinctClosed*[*simp*]:
fixes p :: *name prm*
and xs :: $'a$::*pt-name list*

shows $distinct\ (p \cdot xs) = distinct\ xs$
⟨*proof*⟩

lemma *lengthEqvt*[*eqvt*]:
fixes p :: *name prm*
and xs :: $'a$::*pt-name list*

shows $p \cdot (length\ xs) = length\ (p \cdot xs)$
⟨*proof*⟩

lemma *lengthClosed*[*simp*]:
fixes p :: *name prm*
and xs :: $'a$::*pt-name list*

shows $length\ (p \cdot xs) = length\ xs$
⟨*proof*⟩

lemma *subsetEqvt*[*eqvt*]:
fixes p :: *name prm*
and S :: $('a$::*pt-name*$)\ set$
and T :: $('a$::*pt-name*$)\ set$

shows $(p \cdot (S \subseteq T)) = ((p \cdot S) \subseteq (p \cdot T))$
⟨*proof*⟩

lemma *subsetClosed*[*simp*]:
fixes p :: *name prm*
and S :: $('a$::*pt-name*$)\ set$
and T :: $('a$::*pt-name*$)\ set$

shows $((p \cdot S) \subseteq (p \cdot T)) = (S \subseteq T)$
⟨*proof*⟩

lemma *subsetClosed'*[*simp*]:
fixes p :: *name prm*
and $xvec$ :: *name list*
and P :: $'a$::*fs-name*

shows $(set\ (p \cdot xvec) \subseteq supp\ (p \cdot P)) = (set\ xvec \subseteq supp\ P)$
⟨*proof*⟩

lemma *memEqvt*[*eqvt*]:
  fixes p :: *name prm*
  and x :: $'a$::*pt-name*
  and xs :: ($'a$::*pt-name*) *list*

  shows $(p \cdot (x\ mem\ xs)) = ((p \cdot x)\ mem\ (p \cdot xs))$
⟨*proof*⟩

lemma *memClosed*[*simp*]:
  fixes p :: *name prm*
  and x :: $'a$::*pt-name*
  and xs :: ($'a$::*pt-name*) *list*

  shows $(p \cdot x)\ mem\ (p \cdot xs) = (x\ mem\ xs)$
⟨*proof*⟩

lemma *memClosed'*[*simp*]:
  fixes p :: *name prm*
  and x :: $'a$::*pt-name*
  and y :: $'b$::*pt-name*
  and xs :: ($'a \times\ 'b$) *list*

  shows $((p \cdot x,\ p \cdot y)\ mem\ (p \cdot xs)) = ((x,\ y)\ mem\ xs)$
⟨*proof*⟩

lemma *freshPerm*:
  fixes x :: *name*
  and p :: *name prm*

  assumes $x \sharp p$

  shows $p \cdot x = x$
⟨*proof*⟩

lemma *freshChainPermSimp*:
  fixes *xvec* :: *name list*
  and p :: *name prm*

  assumes *xvec* $\sharp*$ p

  shows $p \cdot xvec = xvec$
  and $rev\ p \cdot xvec = xvec$
⟨*proof*⟩

lemma *freshChainAppend*[*simp*]:
  fixes *xvec* :: *name list*
  and *yvec* :: *name list*
  and C :: $'a$::*fs-name*

shows $(xvec@yvec) \sharp* C = ((xvec \sharp* C) \wedge (yvec \sharp* C))$
⟨*proof*⟩

lemma *subsetFresh*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *C* :: *'d::fs-name*

assumes *set xvec* ⊆ *set yvec*
and *yvec* $\sharp*$ *C*

shows *xvec* $\sharp*$ *C*
⟨*proof*⟩

lemma *distinctPermCancel*[*simp*]:
fixes *p* :: *name prm*
and *T* :: *'a::pt-name*

assumes *distinctPerm p*

shows $(p \cdot (p \cdot T)) = T$
⟨*proof*⟩

fun *composePerm* :: *name list* ⇒ *name list* ⇒ *name prm*
where
Base: *composePerm* [] [] = []
| *Step*: *composePerm* (*x*#*xs*) (*y*#*ys*) = (*x*, *y*)#(*composePerm xs ys*)
| *Empty*: *composePerm* - -= []

lemma *composePermInduct*[*consumes 1*, *case-names cBase cStep*]:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *P* :: *name list* ⇒ *name list* ⇒ *bool*

assumes *L*: *length xvec* = *length yvec*
and *rBase*: *P* [] []
and *rStep*: $\bigwedge x\ xvec\ y\ yvec.\ [\![length\ xvec = length\ yvec;\ P\ xvec\ yvec]\!] \Longrightarrow P\ (x\ \#\ xvec)\ (y\ \#\ yvec)$

shows *P xvec yvec*
⟨*proof*⟩

lemma *composePermEqvt*[*eqvt*]:
fixes *p* :: *name prm*
and *xvec* :: *name list*
and *yvec* :: *name list*

shows $(p \cdot (composePerm\ xvec\ yvec)) = composePerm\ (p \cdot xvec)\ (p \cdot yvec)$

⟨*proof*⟩

abbreviation
composePermJudge (‹[- -] ∙$_v$ -› [*80*, *80*, *80*] *80*) **where** [*xvec yvec*] ∙$_v$ *p* ≡ (*composePerm xvec yvec*) ∙ *p*

abbreviation
composePermInvJudge (‹[- -]$^-$ ∙$_v$ -› [*80*, *80*, *80*] *80*) **where** [*xvec yvec*]$^-$ ∙$_v$ *p* ≡ (*rev* (*composePerm xvec yvec*)) ∙ *p*

lemma *permChainSimps*[*simp*]:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *perm* :: *name prm*
and *p* :: *'a*::*pt-name*

shows ((*composePerm xvec yvec*) @ *perm*) ∙ *p* = [*xvec yvec*] ∙$_v$ (*perm* ∙ *p*)
⟨*proof*⟩

lemma *permChainEqvt*[*eqvt*]:
fixes *p* :: *name prm*
and *xvec* :: *name list*
and *yvec* :: *name list*
and *x* :: *'a*::*pt-name*

shows (*p* ∙ ([*xvec yvec*] ∙$_v$ *x*)) = [(*p* ∙ *xvec*) (*p* ∙ *yvec*)] ∙$_v$ (*p* ∙ *x*)
and (*p* ∙ ([*xvec yvec*]$^-$ ∙$_v$ *x*)) = [(*p* ∙ *xvec*) (*p* ∙ *yvec*)]$^-$ ∙$_v$ (*p* ∙ *x*)
⟨*proof*⟩

lemma *permChainBij*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *p* :: *'a*::*pt-name*
and *q* :: *'a*::*pt-name*

assumes *length xvec* = *length yvec*

shows (([*xvec yvec*] ∙$_v$ *p*) = ([*xvec yvec*] ∙$_v$ *q*)) = (*p* = *q*)
and (([*xvec yvec*]$^-$ ∙$_v$ *p*) = ([*xvec yvec*]$^-$ ∙$_v$ *q*)) = (*p* = *q*)
⟨*proof*⟩

lemma *permChainAppend*:
fixes *xvec1* :: *name list*
and *yvec1* :: *name list*
and *xvec2* :: *name list*
and *yvec2* :: *name list*
and *p* :: *'a*::*pt-name*

assumes *length xvec1* = *length yvec1*

shows $([(xvec1@xvec2)\ (yvec1@yvec2)] \cdot_v p) = [xvec1\ yvec1] \cdot_v [xvec2\ yvec2] \cdot_v p$
and $([(xvec1@xvec2)\ (yvec1@yvec2)]^- \cdot_v p) = [xvec2\ yvec2]^- \cdot_v [xvec1\ yvec1]^- \cdot_v p$
⟨*proof*⟩

lemma *calcChainAtom*:
fixes $xvec :: name\ list$
and $yvec :: name\ list$
and $x :: name$

assumes $length\ xvec = length\ yvec$
and $x \sharp xvec$
and $x \sharp yvec$

shows $[xvec\ yvec] \cdot_v x = x$
⟨*proof*⟩

lemma *calcChainAtomRev*:
fixes $xvec :: name\ list$
and $yvec :: name\ list$
and $x :: name$

assumes $length\ xvec = length\ yvec$
and $x \sharp xvec$
and $x \sharp yvec$

shows $[xvec\ yvec]^- \cdot_v x = x$
⟨*proof*⟩

lemma *permChainFresh*[*simp*]:
fixes $x :: name$
and $xvec :: name\ list$
and $yvec :: name\ list$
and $p :: 'a::pt\text{-}name$

assumes $x \sharp xvec$
and $x \sharp yvec$
and $length\ xvec = length\ yvec$

shows $x \sharp [xvec\ yvec] \cdot_v p = x \sharp p$
and $x \sharp [xvec\ yvec]^- \cdot_v p = x \sharp p$
⟨*proof*⟩

lemma *chainFreshFresh*:
fixes $x :: name$
and $y :: name$
and $xvec :: name\ list$

and p :: $'a$::*pt-name*

assumes $x \sharp xvec$
and $y \sharp xvec$

shows $xvec \sharp* ([(x, y)] \cdot p) = (xvec \sharp* p)$
⟨*proof*⟩

lemma *permChainFreshFresh*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and p :: $'a$::*pt-name*

assumes $xvec \sharp* p$
and $yvec \sharp* p$
and *length xvec* = *length yvec*

shows $[xvec\ yvec] \cdot_v p = p$
and $[xvec\ yvec]^- \cdot_v p = p$
⟨*proof*⟩

lemma *setFresh*[*simp*]:
fixes x :: *name*
and *xvec* :: *name list*

shows $x \notin set\ xvec = x \sharp xvec$
⟨*proof*⟩

lemma *calcChain*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*

assumes $yvec \sharp* xvec$
and *length xvec* = *length yvec*
and *distinct xvec*
and *distinct yvec*

shows $[xvec\ yvec] \cdot_v xvec = yvec$
⟨*proof*⟩

lemma *freshChainPerm*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and x :: *name*
and C :: $'a$::*pt-name*

assumes *length xvec* = *length yvec*
and $yvec \sharp* C$
and $xvec \sharp* yvec$

and $x\ mem\ xvec$
and $distinct\ yvec$

shows $x \sharp [xvec\ yvec] \cdot_v C$
⟨*proof*⟩

lemma *memFreshSimp*[*simp*]:
fixes $y :: name$
and $yvec :: name\ list$

shows $(\neg(y\ mem\ yvec)) = y \sharp yvec$
⟨*proof*⟩

lemma *freshChainPerm′*:
fixes $xvec :: name\ list$
and $yvec :: name\ list$
and $p :: 'a::pt\text{-}name$

assumes $length\ xvec = length\ yvec$
and $yvec \sharp* p$
and $xvec \sharp* yvec$
and $distinct\ yvec$

shows $xvec \sharp* ([xvec\ yvec] \cdot_v p)$
⟨*proof*⟩

lemma *permSym*:
fixes $x :: name$
and $y :: name$
and $xvec :: name\ list$
and $yvec :: name\ list$
and $p :: 'a::pt\text{-}name$

assumes $x \sharp xvec$
and $x \sharp yvec$
and $y \sharp xvec$
and $y \sharp yvec$
and $length\ xvec = length\ yvec$

shows $([(x, y)] \cdot [xvec\ yvec] \cdot_v p) = [xvec\ yvec] \cdot_v [(x, y)] \cdot p$
⟨*proof*⟩

lemma *distinctPermClosed*[*simp*]:
fixes $p :: name\ prm$
and $q :: name\ prm$

assumes $distinctPerm\ p$

shows $distinctPerm(q \cdot p)$

⟨*proof*⟩

lemma *freshStarSimps*:
fixes x :: *name*
and Xs :: *name set*
and Ys :: *name set*
and C :: *'a::fs-name*
and p :: *name prm*

assumes *set* $p \subseteq Xs \times Ys$
and $Xs \sharp* x$
and $Ys \sharp* x$

shows $x \sharp (p \cdot C) = x \sharp C$
⟨*proof*⟩

lemma *freshStarChainSimps*:
fixes *xvec* :: *name list*
and Xs :: *name set*
and Ys :: *name set*
and C :: *'a::fs-name*
and p :: *name prm*

assumes *set* $p \subseteq Xs \times Ys$
and $Xs \sharp* xvec$
and $Ys \sharp* xvec$

shows $xvec \sharp* (p \cdot C) = xvec \sharp* C$
⟨*proof*⟩

lemma *permStarFresh*:
fixes *xvec* :: *name list*
and p :: *name prm*
and T :: *'a::pt-name*

assumes $xvec \sharp* p$

shows $xvec \sharp* (p \cdot T) = xvec \sharp* T$
⟨*proof*⟩

lemma *swapStarFresh*:
fixes x :: *name*
and p :: *name prm*
and T :: *'a::pt-name*

assumes $x \sharp p$

shows $x \sharp (p \cdot T) = x \sharp T$
⟨*proof*⟩

lemmas *freshChainSimps* = *freshStarSimps freshStarChainSimps permStarFresh swapStarFresh chainFreshFresh freshPerm subsetFresh*

lemma *freshAlphaPerm*:
  fixes *xvec* :: *name list*
  and *Xs* :: *name set*
  and *Ys* :: *name set*
  and *p* :: *name prm*

  assumes *S*: *set p* ⊆ *Xs* × *Ys*
  and *Xs* ♯* *xvec*
  and *Ys* ♯* *xvec*

  shows *xvec* ♯* *p*
⟨*proof*⟩

lemma *freshAlphaSwap*:
  fixes *x* :: *name*
  and *Xs* :: *name set*
  and *Ys* :: *name set*
  and *p* :: *name prm*

  assumes *S*: *set p* ⊆ *Xs* × *Ys*
  and *Xs* ♯* *x*
  and *Ys* ♯* *x*

  shows *x* ♯ *p*
⟨*proof*⟩

lemma *setToListFresh*[*simp*]:
  fixes *xvec* :: *name list*
  and *C* :: *'a*::*fs-name*
  and *yvec* :: *name list*
  and *Xs* :: *name set*
  and *x* :: *name*

  shows *xvec* ♯* (*set yvec*) = *xvec* ♯* *yvec*
  and *Xs* ♯* (*set yvec*) = *Xs* ♯* *yvec*
  and *x* ♯ (*set yvec*) = *x* ♯ *yvec*
  and *set xvec* ♯* *Xs* = *xvec* ♯* *Xs*
⟨*proof*⟩

end

theory *Subst-Term*
  imports *Chain*
begin

locale *substType* =
  fixes *subst* :: *'a*::*fs-name* ⇒ *name list* ⇒ *'b*::*fs-name list* ⇒ *'a* (‹-[-::=-]› [*80*, *80* ,*80*] *130*)

  assumes *eq*[*eqvt*]: ⋀*p*::*name prm*. (*p* ∙ (*M*[*xvec*::=*Tvec*])) = ((*p* ∙ *M*)[(*p* ∙ *xvec*)::=(*p* ∙ *Tvec*)])

  and *subst3*: ⋀*xvec Tvec T x*. ⟦*length xvec* = *length Tvec*; *distinct xvec*; *set*(*xvec*) ⊆ *supp*(*T*); (*x*::*name*) ♯ *T*[*xvec*::=*Tvec*]⟧ ⟹ *x* ♯ *Tvec*

  and *renaming*: ⋀*xvec Tvec p T*. ⟦*length xvec* = *length Tvec*; (*set p*) ⊆ *set xvec* × *set* (*p* ∙ *xvec*);
distinctPerm p; (*p* ∙ *xvec*) ♯* *T*⟧ ⟹
T[*xvec*::=*Tvec*] = (*p* ∙ *T*)[(*p* ∙ *xvec*)::=*Tvec*]
begin

lemma *suppSubst*:
  fixes *M* :: *'a*
  and *xvec* :: *name list*
  and *Tvec* :: *'b list*

  shows (*supp*(*M*[*xvec*::=*Tvec*])::*name set*) ⊆ ((*supp M*) ∪ (*supp xvec*) ∪ (*supp Tvec*))
⟨*proof*⟩

lemma *subst2*[*intro*]:
  fixes *x* :: *name*
  and *M* :: *'a*
  and *xvec* :: *name list*
  and *Tvec* :: *'b list*

  assumes *x* ♯ *M*
  and *x* ♯ *xvec*
  and *x* ♯ *Tvec*

  shows *x* ♯ *M*[*xvec*::=*Tvec*]
⟨*proof*⟩

lemma *subst2Chain*[*intro*]:
  fixes *yvec* :: *name list*
  and *M* :: *'a*
  and *xvec* :: *name list*
  and *Tvec* :: *'b list*

  assumes *yvec* ♯* *M*
  and *yvec* ♯* *xvec*
  and *yvec* ♯* *Tvec*

  shows *yvec* ♯* *M*[*xvec*::=*Tvec*]

⟨*proof*⟩

lemma *fs*[*simp*]: *finite* ((*supp subst*)::*name set*)
⟨*proof*⟩

lemma *subst3Chain*:
  fixes *xvec* :: *name list*
  and *Tvec* :: *'b list*
  and *Xs* :: *name set*
  and *T* :: *'a*

  assumes *length xvec* = *length Tvec*
  and *distinct xvec*
  and *set xvec* ⊆ *supp T*
  and *Xs* ♯* *T*[*xvec*::=*Tvec*]

  shows *Xs* ♯* *Tvec*
⟨*proof*⟩

lemma *subst4Chain*:
  fixes *xvec* :: *name list*
  and *Tvec* :: *'b list*
  and *T* :: *'a*

  assumes *length xvec* = *length Tvec*
  and *distinct xvec*
  and *xvec* ♯* *Tvec*

  shows *xvec* ♯* *T*[*xvec*::=*Tvec*]
⟨*proof*⟩

definition *seqSubst* :: *'a* ⇒ (*name list* × *'b list*) *list* ⇒ *'a* (‹-[<->]› [*80*, *80*] *130*)
  where *M*[<*σ*>] ≡ *foldl* (λ*N*. λ(*xvec*, *Tvec*). *N*[*xvec*::=*Tvec*]) *M σ*

lemma *seqSubstNil*[*simp*]:
  seqSubst M [] = *M*
⟨*proof*⟩

lemma *seqSubstCons*[*simp*]:
  shows *seqSubst M* ((*xvec*, *Tvec*)#*σ*) = *seqSubst*(*M*[*xvec*::=*Tvec*]) *σ*
  ⟨*proof*⟩

lemma *seqSubstTermAppend*[*simp*]:
  shows *seqSubst M* (*σ*@*σ'*) = *seqSubst* (*seqSubst M σ*) *σ'*
⟨*proof*⟩

definition *wellFormedSubst* :: ((*'d*::*fs-name*) *list* × (*'e*::*fs-name*) *list*) *list* ⇒ *bool*
where *wellFormedSubst σ* = ((*filter* (λ(*xvec*, *Tvec*). ¬(*length xvec* = *length Tvec* ∧ *distinct xvec*)) *σ*) = [])

lemma *wellFormedSubstEqvt*[*eqvt*]:
fixes σ :: (('*d*::*fs-name*) *list* × ('*e*::*fs-name*) *list*) *list*
and p :: *name prm*

shows $p \cdot (wellFormedSubst\ \sigma) = wellFormedSubst(p \cdot \sigma)$
⟨*proof*⟩

lemma *wellFormedSimp*[*simp*]:
fixes σ :: (('*d*::*fs-name*) *list* × ('*e*::*fs-name*) *list*) *list*
and p :: *name prm*

shows $wellFormedSubst(p \cdot \sigma) = wellFormedSubst\ \sigma$
⟨*proof*⟩

lemma *wellFormedNil*[*simp*]:
$wellFormedSubst\ []$
⟨*proof*⟩

lemma *wellFormedCons*[*simp*]:
shows $wellFormedSubst((xvec, Tvec)\#\sigma) = (length\ xvec = length\ Tvec \wedge distinct\ xvec \wedge wellFormedSubst\ \sigma)$
⟨*proof*⟩

lemma *wellFormedAppend*[*simp*]:
fixes σ :: (('*d*::*fs-name*) *list* × ('*e*::*fs-name*) *list*) *list*
and σ' :: (('*d*::*fs-name*) *list* × ('*e*::*fs-name*) *list*) *list*

shows $wellFormedSubst(\sigma@\sigma') = (wellFormedSubst\ \sigma \wedge wellFormedSubst\ \sigma')$
⟨*proof*⟩

lemma *seqSubst2*[*intro*]:
fixes σ :: (*name list* × '*b list*) *list*
and T :: '*a*
and x :: *name*

assumes $x \sharp \sigma$
and $x \sharp T$

shows $x \sharp T[<\sigma>]$
⟨*proof*⟩

lemma *seqSubst2Chain*[*intro*]:
fixes σ :: (*name list* × '*b list*) *list*
and T :: '*a*
and *xvec* :: *name list*

assumes $xvec \sharp* \sigma$
and $xvec \sharp* T$

shows *xvec* $\sharp*$ *T*[<σ>]
⟨*proof*⟩

end

end

theory *Agent*
imports *Subst-Term*
begin

nominal-datatype ('*term*, '*assertion*, '*condition*) *psi* =
PsiNil (‹**0**› *190*)

| *Output* '*term*::*fs-name* '*term* ('*term*, '*assertion*::*fs-name*, '*condition*::*fs-name*) *psi*
(‹-⟨-⟩.-› [*120*, *120*, *110*] *110*)
| *Input* '*term* ('*term*, '*assertion*, '*condition*) *input* (‹-(-› [*120*, *120*] *110*)
| *Case* (('*term*, '*assertion*, '*condition*) *psiCase*) (‹*Case* -› [*120*] *120*)
| *Par* ('*term*, '*assertion*, '*condition*) *psi* ('*term*, '*assertion*, '*condition*) *psi* (**infixl** ‹‖› *90*)
| *Res* «*name*»(('*term*, '*assertion*, '*condition*) *psi*) (‹(*ν*-)-› [*120*, *120*] *110*)
| *Assert* '*assertion* (‹{-}› [*120*] *120*)
| *Bang* ('*term*, '*assertion*, '*condition*) *psi* (‹!-› [*110*] *110*)

and ('*term*, '*assertion*, '*condition*) *input* =
Trm '*term* (('*term*, '*assertion*, '*condition*) *psi*) (‹)-.-› [*130*, *130*] *130*)
| *Bind* «*name*»(('*term*, '*assertion*, '*condition*) *input*) (‹*ν*--› [*120*, *120*] *120*)

and ('*term*, '*assertion*, '*condition*) *psiCase* =
EmptyCase (‹⊥$_c$› *120*)
| *Cond* '*condition* (('*term*, '*assertion*, '*condition*) *psi*)
(('*term*, '*assertion*, '*condition*) *psiCase*) (‹□ - ⇒ - - › [*120*, *120*, *120*] *120*)

lemma *psiFreshSet*[*simp*]:
fixes *X* :: *name set*
and *M* :: '*a*::*fs-name*
and *N* :: '*a*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *I* :: ('*a*, '*b*, '*c*) *input*

and $C :: ('a, 'b, 'c)\ psiCase$
and $Q :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $\Psi :: 'b$
and $\Phi :: 'c$

shows $X \sharp* (M\langle N\rangle.P) = (X \sharp* M \wedge X \sharp* N \wedge X \sharp* P)$
and $X \sharp* M(\!|I = (X \sharp* M \wedge X \sharp* I)$
and $X \sharp* Case\ C = X \sharp* C$
and $X \sharp* (P \parallel Q) = (X \sharp* P \wedge X \sharp* Q)$
and $X \sharp* (\!|\nu x|\!)P = (X \sharp* [x].P)$
and $X \sharp* \{\!|\Psi|\!\} = X \sharp* \Psi$
and $X \sharp* !P = X \sharp* P$
and $X \sharp* \mathbf{0}$
and $X \sharp* Trm\ N\ P = (X \sharp* N \wedge X \sharp* P)$
and $X \sharp* Bind\ x\ I = X \sharp* ([x].I)$

and $X \sharp* \bot_c$
and $X \sharp* \Box\ \Phi \Rightarrow P\ C = (X \sharp* \Phi \wedge X \sharp* P \wedge X \sharp* C)$
⟨*proof*⟩

lemma *psiFreshVec*[*simp*]:
fixes *xvec* :: *name list*

shows $xvec \sharp* (M\langle N\rangle.P) = (xvec \sharp* M \wedge xvec \sharp* N \wedge xvec \sharp* P)$
and $xvec \sharp* M(\!|I = (xvec \sharp* M \wedge xvec \sharp* I)$
and $xvec \sharp* Case\ C = xvec \sharp* C$
and $xvec \sharp* (P \parallel Q) = (xvec \sharp* P \wedge xvec \sharp* Q)$
and $xvec \sharp* (\!|\nu x|\!)P = (xvec \sharp* [x].P)$
and $xvec \sharp* \{\!|\Psi|\!\} = xvec \sharp* \Psi$
and $xvec \sharp* !P = xvec \sharp* P$
and $xvec \sharp* \mathbf{0}$

and $xvec \sharp* Trm\ N\ P = (xvec \sharp* N \wedge xvec \sharp* P)$
and $xvec \sharp* Bind\ x\ I = xvec \sharp* ([x].I)$

and $xvec \sharp* \bot_c$
and $xvec \sharp* \Box\ \Phi \Rightarrow P\ C = (xvec \sharp* \Phi \wedge xvec \sharp* P \wedge xvec \sharp* C)$
⟨*proof*⟩

fun *psiCases* :: ('*c*::*fs-name* × ('*a*::*fs-name*, '*b*::*fs-name*, '*c*) *psi*) *list* ⇒ ('*a*, '*b*, '*c*) *psiCase*
where
base: $psiCases\ [] = \bot_c$
| *step*: $psiCases\ ((\Phi, P)\#xs) = Cond\ \Phi\ P\ (psiCases\ xs)$

lemma *psiCasesEqvt*[*eqvt*]:
fixes p :: *name prm*
and *Cs* :: ('*c*::*fs-name* × ('*a*::*fs-name*, '*b*::*fs-name*, '*c*) *psi*) *list*

shows $(p \cdot (psiCases\ Cs)) = psiCases(p \cdot Cs)$
⟨*proof*⟩

lemma *psiCasesFresh*[*simp*]:
fixes x :: *name*
and Cs :: $('c::fs\text{-}name \times ('a::fs\text{-}name, 'b::fs\text{-}name, 'c)\ psi)\ list$

shows $x \sharp psiCases\ Cs = x \sharp Cs$
⟨*proof*⟩

lemma *psiCasesFreshChain*[*simp*]:
fixes *xvec* :: *name list*
and Cs :: $('c::fs\text{-}name \times ('a::fs\text{-}name, 'b::fs\text{-}name, 'c)\ psi)\ list$
and Xs :: *name set*

shows $(xvec \sharp* psiCases\ Cs) = xvec \sharp* Cs$
and $(Xs \sharp* psiCases\ Cs) = Xs \sharp* Cs$
⟨*proof*⟩

abbreviation
$psiCasesJudge\ (\langle Cases \text{-}\rangle\ [80]\ 80)$ **where** $Cases\ Cs \equiv Case(psiCases\ Cs)$

primrec *resChain* :: *name list* ⇒ $('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$ ⇒ $('a, 'b, 'c)\ psi$ **where**
base: $resChain\ []\ P = P$
| *step*: $resChain\ (x\#xs)\ P = (\!|\nu x|\!)(resChain\ xs\ P)$

notation $resChain\ (\langle(\!|\nu*\text{-}|\!)\text{-}\rangle\ [80, 80]\ 80)$

lemma *resChainEqvt*[*eqvt*]:
fixes *perm* :: *name prm*
and *lst* :: *name list*
and P :: $('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $perm \cdot ((\!|\nu*xvec|\!)P) = (\!|\nu*(perm \cdot xvec)|\!)(perm \cdot P)$
⟨*proof*⟩

lemma *resChainSupp*:
fixes *xvec* :: *name list*
and P :: $('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $supp((\!|\nu*xvec|\!)P) = (supp\ P) - set\ xvec$
⟨*proof*⟩

lemma *resChainFresh*:
fixes x :: *name*
and *xvec* :: *name list*
and P :: $('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $x \sharp (\!|\nu{*}xvec|\!)P = (x \in set\ xvec \vee x \sharp P)$
⟨*proof*⟩

lemma *resChainFreshSet*:
fixes Xs :: *name set*
and $xvec$ :: *name list*
and $yvec$ :: *name list*
and P :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

shows $Xs \sharp* (\!|\nu{*}xvec|\!)P) = (\forall x{\in}Xs.\ x \in set\ xvec \vee x \sharp P)$
and $yvec \sharp* (\!|\nu{*}xvec|\!)P) = (\forall x{\in}(set\ yvec).\ x \in set\ xvec \vee x \sharp P)$
⟨*proof*⟩

lemma *resChainFreshSimps*[*simp*]:
fixes Xs :: *name set*
and $xvec$ :: *name list*
and P :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and $yvec$ :: *name list*

shows $Xs \sharp* xvec \Longrightarrow Xs \sharp* (\!|\nu{*}xvec|\!)P) = (Xs \sharp* P)$
and $yvec \sharp* xvec \Longrightarrow yvec \sharp* (\!|\nu{*}xvec|\!)P) = (yvec \sharp* P)$
and $xvec \sharp* (\!|\nu{*}xvec|\!)P)$
⟨*proof*⟩

lemma *resChainAlpha*:
fixes p :: *name prm*
and $xvec$ :: *name list*
and P :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

assumes *xvecFreshP*: $(p \cdot xvec) \sharp* P$
and S: $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$

shows $(\!|\nu{*}xvec|\!)P = (\!|\nu{*}(p \cdot xvec)|\!)(p \cdot P)$
⟨*proof*⟩

lemma *resChainAppend*:
fixes $xvec$ :: *name list*
and $yvec$ :: *name list*
and P :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

shows $(\!|\nu{*}(xvec@yvec)|\!)P = (\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)P)$
⟨*proof*⟩

lemma *resChainSimps*[*dest*]:
fixes $xvec$ :: *name list*
and P :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and Q :: (*'a*, *'b*, *'c*) *psi*
and P' :: (*'a*, *'b*, *'c*) *psi*

and Q' :: ('a, 'b, 'c) *psi*

shows $((\!|\nu * xvec|\!)(P \parallel Q) = P' \parallel Q') \Longrightarrow (P = P' \wedge Q = Q')$
and $(P \parallel Q = (\!|\nu * xvec|\!)(P' \parallel Q')) \Longrightarrow (P = P' \wedge Q = Q')$
⟨*proof*⟩

primrec *inputChain* :: *name list* ⇒ 'a::*fs-name* ⇒ ('a, 'b::*fs-name*, 'c::*fs-name*) *psi* ⇒ ('a, 'b, 'c) *input* **where**
base: *inputChain* [] *N P* = (|*N*).*P*
| *step*: *inputChain* (*x*#*xs*) *N P* = ν *x* (*inputChain xs N P*)

abbreviation
inputChainJudge (‹-(|λ*- -|).-› [80, 80, 80, 80] 80) **where** $M(\!|\lambda * xvec\ N|\!).P \equiv M(\!|(inputChain\ xvec\ N\ P)$

lemma *inputChainEqvt*[*eqvt*]:
fixes *p* :: *name prm*
and *xvec* :: *name list*
and *N* :: 'a::*fs-name*
and *P* :: ('a, 'b::*fs-name*, 'c::*fs-name*) *psi*

shows $p \cdot (inputChain\ xvec\ N\ P) = inputChain\ (p \cdot xvec)\ (p \cdot N)\ (p \cdot P)$
⟨*proof*⟩

lemma *inputChainFresh*:
fixes *x* :: *name*
and *xvec* :: *name list*
and *N* :: 'a::*fs-name*
and *P* :: ('a, 'b::*fs-name*, 'c::*fs-name*) *psi*

shows $x \sharp (inputChain\ xvec\ N\ P) = (x \in set\ xvec \vee (x \sharp N \wedge x \sharp P))$
⟨*proof*⟩

lemma *inductChainSimps*[*simp*]:
fixes *xvec* :: *name list*
and *N* :: 'a::*fs-name*
and *P* :: ('a, 'b::*fs-name*, 'c::*fs-name*) *psi*

shows $xvec \sharp* (inputChain\ xvec\ N\ P)$
⟨*proof*⟩

lemma *inputChainFreshSet*:
fixes *Xs* :: *name set*
and *xvec* :: *name list*
and *N* :: 'a::*fs-name*
and *P* :: ('a, 'b::*fs-name*, 'c::*fs-name*) *psi*

shows $Xs \sharp* (inputChain\ xvec\ N\ P) = (\forall x \in Xs.\ x \in set\ xvec \vee (x \sharp N \wedge x \sharp P))$
⟨*proof*⟩

lemma *inputChainAlpha*:
fixes p :: *name prm*
and Xs :: *name set*
and Ys :: *name set*

assumes *XsFreshP*: $Xs \sharp* (inputChain\ xvec\ N\ P)$
and *YsFreshN*: $Ys \sharp* N$
and *YsFreshP*: $Ys \sharp* P$
and S: $set\ p \subseteq Xs \times Ys$

shows $(inputChain\ xvec\ N\ P) = (inputChain\ (p \cdot xvec)\ (p \cdot N)\ (p \cdot P))$
⟨*proof*⟩

lemma *inputChainAlpha'*:
fixes p :: *name prm*
and *xvec* :: *name list*
and N :: *'a*::*fs-name*
and P :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

assumes *xvecFreshP*: $(p \cdot xvec) \sharp* P$
and *xvecFreshN*: $(p \cdot xvec) \sharp* N$
and S: $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$

shows $(inputChain\ xvec\ N\ P) = (inputChain\ (p \cdot xvec)\ (p \cdot N)\ (p \cdot P))$
⟨*proof*⟩

lemma *alphaRes*:
fixes M :: *'a*::*fs-name*
and x :: *name*
and P :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and y :: *name*

assumes *yFreshP*: $y \sharp P$

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

lemma *alphaInput*:
fixes x :: *name*
and I :: (*'a*::*fs-name*, *'b*::*fs-name*, *'c*::*fs-name*) *input*
and c :: *name*

assumes *A1*: $c \sharp I$

shows $\nu\ x\ I = \nu\ c([(x, c)] \cdot I)$
⟨*proof*⟩

lemma *inputChainLengthEq*:

fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *M* :: *'a*::*fs-name*
and *P* :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

assumes *length xvec* = *length yvec*
and *xvec* $\sharp*$ *yvec*
and *distinct yvec*
and *yvec* $\sharp*$ *M*
and *yvec* $\sharp*$ *P*

obtains *N Q* **where** *inputChain xvec M P* = *inputChain yvec N Q*
⟨*proof*⟩

lemma *inputChainEq*:
fixes *xvec* :: *name list*
and *M* :: *'a*::*fs-name*
and *P* :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and *yvec* :: *name list*
and *N* :: *'a*
and *Q* :: (*'a*, *'b*, *'c*) *psi*

assumes *inputChain xvec M P* = *inputChain yvec N Q*
and *xvec* $\sharp*$ *yvec*
and *distinct xvec*
and *distinct yvec*

obtains *p* **where** (*set p*) ⊆ (*set xvec*) × *set* (*p* ∙ *xvec*) **and** *distinctPerm p* **and** *yvec* = *p* ∙ *xvec* **and** *N* = *p* ∙ *M* **and** *Q* = *p* ∙ *P*
⟨*proof*⟩

lemma *inputChainEqLength*:
fixes *xvec* :: *name list*
and *M* :: *'a*::*fs-name*
and *P* :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*
and *yvec* :: *name list*
and *N* :: *'a*
and *Q* :: (*'a*, *'b*, *'c*) *psi*

assumes *inputChain xvec M P* = *inputChain yvec N Q*

shows *length xvec* = *length yvec*
⟨*proof*⟩

lemma *alphaInputChain*:
fixes *yvec* :: *name list*
and *xvec* :: *name list*
and *M* :: *'a*::*fs-name*
and *P* :: (*'a*, *'b*::*fs-name*, *'c*::*fs-name*) *psi*

assumes *length xvec = length yvec*
and *yvec ♯* M*
and *yvec ♯* P*
and *yvec ♯* xvec*
and *distinct yvec*

shows *inputChain xvec M P = inputChain yvec ([xvec yvec] ∙$_v$ M) ([xvec yvec] ∙$_v$ P)*
⟨*proof*⟩

lemma *inputChainInject*[*simp*]:

shows *(inputChain xvec M P = inputChain xvec N Q) = ((M = N) ∧ (P = Q))*
⟨*proof*⟩

lemma *alphaInputDistinct*:
fixes *xvec :: name list*
and *M :: 'a::fs-name*
and *P :: ('a, 'b::fs-name, 'c::fs-name) psi*
and *yvec :: name list*
and *N :: 'a*
and *Q :: ('a, 'b, 'c) psi*

assumes *Eq*: *inputChain xvec M P = inputChain yvec N Q*
and *xvecDist*: *distinct xvec*
and *Mem*: *⋀x. x ∈ set xvec ⟹ x ∈ supp M*
and *xvecFreshyvec*: *xvec ♯* yvec*
and *xvecFreshN*: *xvec ♯* N*
and *xvecFreshQ*: *xvec ♯* Q*

shows *distinct yvec*
⟨*proof*⟩

lemma *psiCasesInject*[*simp*]:
fixes *CsP :: ('c::fs-name × ('a::fs-name, 'b::fs-name, 'c) psi) list*
and *CsQ :: ('c × ('a, 'b, 'c) psi) list*

shows *(psiCases CsP = psiCases CsQ) = (CsP = CsQ)*
⟨*proof*⟩

lemma *casesInject*[*simp*]:
fixes *CsP :: ('c::fs-name × ('a::fs-name, 'b::fs-name, 'c) psi) list*
and *CsQ :: ('c × ('a, 'b, 'c) psi) list*

shows *(Cases CsP = Cases CsQ) = (CsP = CsQ)*
⟨*proof*⟩

nominal-primrec

guarded :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psi* ⇒ *bool*
and *guarded'* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *input* ⇒ *bool*
and *guarded''* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psiCase* ⇒ *bool*

where
guarded (**0**) = *True*
| *guarded* (*M*⟨*N*⟩.*P*) = *True*
| *guarded* (*M*(⦇*I*) = *True*
| *guarded* (*Case C*) = *guarded''* *C*
| *guarded* (*P* ∥ *Q*) = ((*guarded P*) ∧ (*guarded Q*))
| *guarded* (⦇*νx*⦈*P*) = (*guarded P*)
| *guarded* ({⦇Ψ⦈}) = *False*
| *guarded* (!*P*) = *guarded P*

| *guarded'* (*Trm M P*) = *False*
| *guarded'* (*ν y I*) = *False*

| *guarded''* (⊥$_c$) = *True*
| *guarded''* (□*φ* ⇒ *P C*) = (*guarded P* ∧ *guarded''* *C*)
⟨*proof*⟩

lemma *guardedEqvt*[*eqvt*]:
fixes *p* :: *name prm*
and *P* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *I* :: ('*a*, '*b*, '*c*) *input*
and *C* :: ('*a*, '*b*, '*c*) *psiCase*

shows (*p* ∙ (*guarded P*)) = *guarded* (*p* ∙ *P*)
and (*p* ∙ (*guarded'* *I*)) = *guarded'* (*p* ∙ *I*)
and (*p* ∙ (*guarded''* *C*)) = *guarded''* (*p* ∙ *C*)
⟨*proof*⟩

lemma *guardedClosed*[*simp*]:
fixes *P* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *p* :: *name prm*

assumes *guarded P*

shows *guarded*(*p* ∙ *P*)
⟨*proof*⟩

locale *substPsi* =
substTerm?: *substType substTerm* +
substAssert?: *substType substAssert* +
substCond?: *substType substCond*

for *substTerm* :: ('*a*::*fs-name*) ⇒ *name list* ⇒ '*a*::*fs-name list* ⇒ '*a*
and *substAssert* :: ('*b*::*fs-name*) ⇒ *name list* ⇒ '*a*::*fs-name list* ⇒ '*b*
and *substCond* :: ('*c*::*fs-name*) ⇒ *name list* ⇒ '*a*::*fs-name list* ⇒ '*c*

begin

nominal-primrec
subs :: ('a::*fs-name*, 'b::*fs-name*, 'c::*fs-name*) *psi* ⇒ *name list* ⇒ 'a *list* ⇒ ('a, 'b, 'c) *psi*
and *subs'* :: ('a::*fs-name*, 'b::*fs-name*, 'c::*fs-name*) *input* ⇒ *name list* ⇒ 'a *list* ⇒ ('a, 'b, 'c) *input*
and *subs''* :: ('a::*fs-name*, 'b::*fs-name*, 'c::*fs-name*) *psiCase* ⇒ *name list* ⇒ 'a *list* ⇒ ('a, 'b, 'c) *psiCase*

where
subs (**0**) *xvec Tvec* = **0**
| (*subs* (*M*⟨*N*⟩.*P*) *xvec Tvec*) = (*substTerm M xvec Tvec*)⟨(*substTerm N xvec Tvec*)⟩.(*subs P xvec Tvec*)
| (*subs* (*M*(⌊*I*) *xvec Tvec*) = (*substTerm M xvec Tvec*)(⌊(*subs' I xvec Tvec*)

| (*subs* (*Case C*) *xvec Tvec*) = (*Case* (*subs'' C xvec Tvec*))
| (*subs* (*P* ∥ *Q*) *xvec Tvec*) = (*subs P xvec Tvec*) ∥ (*subs Q xvec Tvec*)
| ⟦*y* ♯ *xvec*; *y* ♯ *Tvec*⟧ ⟹ (*subs* ((⌊*νy*⌋)*P*) *xvec Tvec*) = (⌊*νy*⌋)(*subs P xvec Tvec*)
| (*subs* (⦃Ψ⦄) *xvec Tvec*) = ⦃(*substAssert* Ψ *xvec Tvec*)⦄
| (*subs* (!*P*) *xvec Tvec*) = !(*subs P xvec Tvec*)

| (*subs'* ((*Trm M P*)::('a::*fs-name*, 'b::*fs-name*, 'c::*fs-name*) *input*) *xvec Tvec*) = (⌋)(*substTerm M xvec Tvec*).(*subs P xvec Tvec*))
| ⟦*y* ♯ *xvec*; *y* ♯ *Tvec*⟧ ⟹ (*subs'* (*ν y I*) *xvec Tvec*) = (*ν y* (*subs' I xvec Tvec*))

| (*subs''* (⊥$_c$::('a::*fs-name*, 'b::*fs-name*, 'c::*fs-name*) *psiCase*) *xvec Tvec*) = ⊥$_c$
| (*subs''* (□Φ ⇒ *P C*) *xvec Tvec*) = (□(*substCond* Φ *xvec Tvec*) ⇒ (*subs P xvec Tvec*) (*subs'' C xvec Tvec*))
⟨*proof*⟩

lemma *substEqvt*[*eqvt*]:
fixes *p* :: *name prm*
and *P* :: ('a, 'b, 'c) *psi*
and *xvec* :: *name list*
and *Tvec* :: 'a *list*
and *I* :: ('a, 'b, 'c) *input*
and *C* :: ('a, 'b, 'c) *psiCase*

shows (*p* ∙ (*subs P xvec Tvec*)) = *subs* (*p* ∙ *P*) (*p* ∙ *xvec*) (*p* ∙ *Tvec*)
and (*p* ∙ (*subs' I xvec Tvec*)) = *subs'* (*p* ∙ *I*) (*p* ∙ *xvec*) (*p* ∙ *Tvec*)
and (*p* ∙ (*subs'' C xvec Tvec*)) = *subs''* (*p* ∙ *C*) (*p* ∙ *xvec*) (*p* ∙ *Tvec*)
⟨*proof*⟩

lemma *subst2*[*intro*]:
fixes *xvec* :: *name list*
and *Tvec* :: 'a *list*
and *x* :: *name*
and *P* :: ('a, 'b, 'c) *psi*

and I :: $('a, 'b, 'c)$ *input*
and C :: $('a, 'b, 'c)$ *psiCase*

assumes $x \sharp Tvec$
and $x \sharp xvec$

shows $x \sharp P \Longrightarrow x \sharp (subs\ P\ xvec\ Tvec)$
and $x \sharp I \Longrightarrow x \sharp (subs'\ I\ xvec\ Tvec)$
and $x \sharp C \Longrightarrow x \sharp (subs''\ C\ xvec\ Tvec)$
⟨*proof*⟩

lemma *subst2Chain*[*intro*]:
fixes $xvec$ :: *name list*
and $Tvec$ :: $'a$ *list*
and Xs :: *name set*
and P :: $('a, 'b, 'c)$ *psi*
and I :: $('a, 'b, 'c)$ *input*
and C :: $('a, 'b, 'c)$ *psiCase*

assumes $Xs \sharp* xvec$
and $Xs \sharp* Tvec$

shows $Xs \sharp* P \Longrightarrow Xs \sharp* (subs\ P\ xvec\ Tvec)$
and $Xs \sharp* I \Longrightarrow Xs \sharp* (subs'\ I\ xvec\ Tvec)$
and $Xs \sharp* C \Longrightarrow Xs \sharp* (subs''\ C\ xvec\ Tvec)$
⟨*proof*⟩

lemma *renaming*:
fixes $xvec$ :: *name list*
and $Tvec$ :: $'a$ *list*
and p :: *name prm*
and P :: $('a, 'b, 'c)$ *psi*
and I :: $('a ,'b, 'c)$ *input*
and C :: $('a, 'b, 'c)$ *psiCase*

assumes $length\ xvec = length\ Tvec$
and $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$
and $distinctPerm\ p$

shows $[\![(p \cdot xvec) \sharp* P]\!] \Longrightarrow (subs\ P\ xvec\ Tvec) = subs\ (p \cdot P)\ (p \cdot xvec)\ Tvec$
and $[\![(p \cdot xvec) \sharp* I]\!] \Longrightarrow (subs'\ I\ xvec\ Tvec) = subs'\ (p \cdot I)\ (p \cdot xvec)\ Tvec$
and $[\![(p \cdot xvec) \sharp* C]\!] \Longrightarrow (subs''\ C\ xvec\ Tvec) = subs''\ (p \cdot C)\ (p \cdot xvec)\ Tvec$
⟨*proof*⟩

lemma *subst4Chain*:
fixes $xvec$ :: *name list*
and $Tvec$ :: $'a$ *list*
and P :: $('a, 'b, 'c)$ *psi*
and I :: $('a, 'b, 'c)$ *input*

and C :: $('a, 'b, 'c)$ *psiCase*

assumes *length xvec* = *length Tvec*
and *distinct xvec*
and *xvec* $\sharp*$ *Tvec*

shows *xvec* $\sharp*$ (*subs P xvec Tvec*)
and *xvec* $\sharp*$ (*subs' I xvec Tvec*)
and *xvec* $\sharp*$ (*subs'' C xvec Tvec*)
⟨*proof*⟩

lemma *guardedSubst*[*simp*]:
fixes P :: $('a, 'b, 'c)$ *psi*
and I :: $('a, 'b, 'c)$ *input*
and C :: $('a, 'b, 'c)$ *psiCase*
and *xvec* :: *name list*
and *Tvec* :: $'a$ *list*

assumes *length xvec* = *length Tvec*
and *distinct xvec*

shows *guarded P* ⟹ *guarded*(*subs P xvec Tvec*)
and *guarded' I* ⟹ *guarded'*(*subs' I xvec Tvec*)
and *guarded'' C* ⟹ *guarded''*(*subs'' C xvec Tvec*)
⟨*proof*⟩

definition *seqSubs* :: $('a, 'b, 'c)$ *psi* ⇒ (*name list* × $'a$ *list*) *list* ⇒ $('a, 'b, 'c)$ *psi* (‹-[<->]› [80, 80] 130)
where $P[<\sigma>] \equiv$ *foldl* ($\lambda Q.\ \lambda$(*xvec*, *Tvec*). *subs Q xvec Tvec*) $P\ \sigma$

definition *seqSubs'* :: $('a, 'b, 'c)$ *input* ⇒ (*name list* × $'a$ *list*) *list* ⇒ $('a, 'b, 'c)$ *input*
where *seqSubs' I* $\sigma \equiv$ *foldl* ($\lambda Q.\ \lambda$(*xvec*, *Tvec*). *subs' Q xvec Tvec*) *I* σ

definition *seqSubs''* :: $('a, 'b, 'c)$ *psiCase* ⇒ (*name list* × $'a$ *list*) *list* ⇒ $('a, 'b, 'c)$ *psiCase*
where *seqSubs''* *C* $\sigma \equiv$ *foldl* ($\lambda Q.\ \lambda$(*xvec*, *Tvec*). *subs'' Q xvec Tvec*) *C* σ

lemma *substInputChain*[*simp*]:
fixes *xvec* :: *name list*
and N :: $'a$
and P :: $('a, 'b, 'c)$ *psi*
and *yvec* :: *name list*
and *Tvec* :: $'a$ *list*

assumes *xvec* $\sharp*$ *yvec*
and *xvec* $\sharp*$ *Tvec*

shows *subs'* (*inputChain xvec N P*) *yvec Tvec* = *inputChain xvec* (*substTerm N*

yvec Tvec) (*subs P yvec Tvec*)
⟨*proof*⟩

fun *caseListSubst* :: ('*c* × ('*a*, '*b*, '*c*) *psi*) *list* ⇒ *name list* ⇒ '*a list* ⇒ ('*c* × ('*a*, '*b*, '*c*) *psi*) *list*
where
caseListSubst [] - - = []
| *caseListSubst* ((*φ*, *P*)#*Cs*) *xvec Tvec* = (*substCond φ xvec Tvec*, (*subs P xvec Tvec*))#(*caseListSubst Cs xvec Tvec*)

lemma *substCases*[*simp*]:
fixes *Cs* :: ('*c* × ('*a*, '*b*, '*c*) *psi*) *list*
and *xvec* :: *name list*
and *Tvec* :: '*a list*

shows *subs* (*Cases Cs*) *xvec Tvec* = *Cases*(*caseListSubst Cs xvec Tvec*)
⟨*proof*⟩

lemma *substCases'*[*simp*]:
fixes *Cs* :: ('*c* × ('*a*, '*b*, '*c*) *psi*) *list*
and *xvec* :: *name list*
and *Tvec* :: '*a list*

shows (*subs''* (*psiCases Cs*) *xvec Tvec*) = *psiCases*(*caseListSubst Cs xvec Tvec*)
⟨*proof*⟩

lemma *seqSubstSimps*[*simp*]:
shows *seqSubs* (**0**) *σ* = **0**
and (*seqSubs* (*M*⟨*N*⟩.*P*) *σ*) = (*substTerm.seqSubst M σ*)⟨(*substTerm.seqSubst N σ*)⟩.(*seqSubs P σ*)
and (*seqSubs* (*M*(⦇*I*) *σ*) = (*substTerm.seqSubst M σ*)(⦇(*seqSubs' I σ*)

and (*seqSubs* (*Case C*) *σ*) = (*Case* (*seqSubs'' C σ*))
and (*seqSubs* (*P* ∥ *Q*) *σ*) = (*seqSubs P σ*) ∥ (*seqSubs Q σ*)
and ⟦*y* ♯ *σ*⟧ ⟹ (*seqSubs* ((⦇*νy*⦈*P*) *σ*) = (⦇*νy*⦈(*seqSubs P σ*)
and (*seqSubs* ({⦇Ψ⦈}) *σ*) = {⦇(*substAssert.seqSubst* Ψ *σ*)⦈}
and (*seqSubs* (!*P*) *σ*) = !(*seqSubs P σ*)

and (*seqSubs'* ((*Trm M P*)::('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *input*) *σ*) = (⦇)(*substTerm.seqSubst M σ*).(*seqSubs P σ*))
and ⟦*y* ♯ *σ*⟧ ⟹ (*seqSubs'* (*ν y I*) *σ*) = (*ν y* (*seqSubs' I σ*))

and (*seqSubs''* (⊥$_c$::('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psiCase*) *σ*) = ⊥$_c$
and (*seqSubs''* (□Φ ⇒ *P C*) *σ*) = (□(*substCond.seqSubst* Φ *σ*) ⇒ (*seqSubs P σ*) (*seqSubs'' C σ*))
⟨*proof*⟩

lemma *seqSubsNil*[*simp*]:
seqSubs P [] = *P*

⟨*proof*⟩

lemma *seqSubsCons*[*simp*]:
shows *seqSubs P* ((*xvec*, *Tvec*)#*σ*) = *seqSubs*(*subs P xvec Tvec*) *σ*
⟨*proof*⟩

lemma *seqSubsTermAppend*[*simp*]:
shows *seqSubs P* (*σ*@*σ*′) = *seqSubs* (*seqSubs P σ*) *σ*′
⟨*proof*⟩

fun *caseListSeqSubst* :: (′*c* × (′*a*, ′*b*, ′*c*) *psi*) *list* ⇒ (*name list* × ′*a list*) *list* ⇒ (′*c* × (′*a*, ′*b*, ′*c*) *psi*) *list*
where
caseListSeqSubst [] - = []
| *caseListSeqSubst* ((*φ*, *P*)#*Cs*) *σ* = (*substCond.seqSubst φ σ*, (*seqSubs P σ*))#(*caseListSeqSubst Cs σ*)

lemma *seqSubstCases*[*simp*]:
fixes *Cs* :: (′*c* × (′*a*, ′*b*, ′*c*) *psi*) *list*
and *σ* :: (*name list* × ′*a list*) *list*

shows *seqSubs* (*Cases Cs*) *σ* = *Cases*(*caseListSeqSubst Cs σ*)
⟨*proof*⟩

lemma *seqSubstCases*′[*simp*]:
fixes *Cs* :: (′*c* × (′*a*, ′*b*, ′*c*) *psi*) *list*
and *σ* :: (*name list* × ′*a list*) *list*

shows (*seqSubs*′′ (*psiCases Cs*) *σ*) = *psiCases*(*caseListSeqSubst Cs σ*)
⟨*proof*⟩

lemma *seqSubstEqvt*[*eqvt*]:
fixes *P* :: (′*a*, ′*b*, ′*c*) *psi*
and *σ* :: (*name list* × ′*a list*) *list*
and *p* :: *name prm*

shows (*p* ∙ (*P*[<*σ*>])) = (*p* ∙ *P*)[<(*p* ∙ *σ*)>]
⟨*proof*⟩

lemma *guardedSeqSubst*:
assumes *guarded P*
and *wellFormedSubst σ*

shows *guarded*(*seqSubs P σ*)
⟨*proof*⟩

end

lemma *inter-eqvt*:

shows $(pi::name\ prm) \cdot ((X::name\ set) \cap Y) = (pi \cdot X) \cap (pi \cdot Y)$
⟨*proof*⟩

lemma *delete-eqvt*:
fixes $p :: name\ prm$
and $X :: name\ set$
and $Y :: name\ set$

shows $p \cdot (X - Y) = (p \cdot X) - (p \cdot Y)$
⟨*proof*⟩

lemma *perm-singleton*[*simp*]:
shows $(p::name\ prm) \cdot \{(x::name)\} = \{p \cdot x\}$
⟨*proof*⟩

end

theory *Frame*
imports *Agent*
begin

lemma *permLength*[*simp*]:
fixes $p :: name\ prm$
and $xvec :: 'a::pt\text{-}name\ list$

shows $length(p \cdot xvec) = length\ xvec$
⟨*proof*⟩

nominal-datatype $'assertion\ frame =$
$FAssert\ 'assertion::fs\text{-}name$
| $FRes\ «name»\ ('assertion\ frame)\ (‹(⦇\nu\text{-}⦈)›\ [80, 80]\ 80)$

primrec $frameResChain :: name\ list \Rightarrow ('a::fs\text{-}name)\ frame \Rightarrow 'a\ frame$ **where**
$base: frameResChain\ []\ F = F$
| $step: frameResChain\ (x\#xs)\ F = (⦇\nu x⦈)(frameResChain\ xs\ F)$

notation $frameResChain\ (‹(⦇\nu*\text{-}⦈)›\ [80, 80]\ 80)$
notation $FAssert\ (‹⟨\varepsilon, \text{-}⟩›\ [80]\ 80)$
abbreviation $FAssertJudge\ (‹⟨\text{-}, \text{-}⟩›\ [80, 80]\ 80)$ **where** $⟨A_F, \Psi_F⟩ \equiv frameResChain\ A_F\ (FAssert\ \Psi_F)$

lemma *frameResChainEqvt*[*eqvt*]:
fixes $perm :: name\ prm$
and $lst :: name\ list$
and $F :: 'a::fs\text{-}name\ frame$

shows $perm \cdot ((⦇\nu*xvec⦈)F) = (⦇\nu*(perm \cdot xvec)⦈)(perm \cdot F)$
⟨*proof*⟩

lemma *frameResChainFresh*:
fixes x :: *name*
and *xvec* :: *name list*
and F :: *'a::fs-name frame*

shows $x \sharp (\!|\nu * xvec|\!) F = (x \in set\ xvec \vee x \sharp F)$
⟨*proof*⟩

lemma *frameResChainFreshSet*:
fixes *Xs* :: *name set*
and *xvec* :: *name list*
and F :: *'a::fs-name frame*

shows $Xs \sharp* (\!|\nu * xvec|\!) F = (\forall x \in Xs.\ x \in set\ xvec \vee x \sharp F)$
⟨*proof*⟩

lemma *frameChainAlpha*:
fixes p :: *name prm*
and *xvec* :: *name list*
and F :: *'a::fs-name frame*

assumes *xvecFreshF*: $(p \cdot xvec) \sharp* F$
and S: $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$

shows $(\!|\nu * xvec|\!) F = (\!|\nu * (p \cdot xvec)|\!)(p \cdot F)$
⟨*proof*⟩

lemma *frameChainAlpha'*:
fixes p :: *name prm*
and A_P :: *name list*
and Ψ_P :: *'a::fs-name*

assumes $(p \cdot A_P) \sharp* \Psi_P$
and S: $set\ p \subseteq set\ A_P \times set\ (p \cdot A_P)$

shows $\langle A_P, \Psi_P \rangle = \langle (p \cdot A_P), p \cdot \Psi_P \rangle$
⟨*proof*⟩

lemma *alphaFrameRes*:
fixes x :: *name*
and F :: *'a::fs-name frame*
and y :: *name*

assumes $y \sharp F$

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

lemma *frameChainAppend*:

fixes $xvec :: name\ list$
and $yvec :: name\ list$
and $F :: 'a::fs\text{-}name\ frame$

shows $(\!|\nu * (xvec @ yvec)|\!) F = (\!|\nu * xvec|\!)((\!|\nu * yvec|\!) F)$
⟨*proof*⟩

lemma *frameChainEqLength*:
fixes $xvec :: name\ list$
and $\Psi :: 'a::fs\text{-}name$
and $yvec :: name\ list$
and $\Psi' :: 'a::fs\text{-}name$

assumes $\langle xvec, \Psi\rangle = \langle yvec, \Psi'\rangle$

shows $length\ xvec = length\ yvec$
⟨*proof*⟩

lemma *frameEqFresh*:
fixes $F :: ('a::fs\text{-}name)\ frame$
and $G :: 'a\ frame$
and $x :: name$
and $y :: name$

assumes $(\!|\nu x|\!) F = (\!|\nu y|\!) G$
and $x \sharp F$

shows $y \sharp G$
⟨*proof*⟩

lemma *frameEqSupp*:
fixes $F :: ('a::fs\text{-}name)\ frame$
and $G :: 'a\ frame$
and $x :: name$
and $y :: name$

assumes $(\!|\nu x|\!) F = (\!|\nu y|\!) G$
and $x \in supp\ F$

shows $y \in supp\ G$
⟨*proof*⟩

lemma *frameChainEqSuppEmpty*[*dest*]:
fixes $xvec :: name\ list$
and $\Psi :: 'a::fs\text{-}name$
and $yvec :: name\ list$
and $\Psi' :: 'a::fs\text{-}name$

assumes $\langle xvec, \Psi\rangle = \langle yvec, \Psi'\rangle$

and *supp* $\Psi = (\{\}::name\ set)$

shows $\Psi = \Psi'$
⟨*proof*⟩

lemma *frameChainEq*:
fixes *xvec* :: *name list*
and Ψ :: $'a::fs\text{-}name$
and *yvec* :: *name list*
and Ψ' :: $'a::fs\text{-}name$

assumes $\langle xvec, \Psi\rangle = \langle yvec, \Psi'\rangle$
and *xvec* $\sharp*$ *yvec*

obtains *p* **where** (*set p*) $\subseteq$ (*set xvec*) $\times$ *set* (*yvec*) **and** *distinctPerm p* **and** Ψ' $= p \cdot \Psi$
⟨*proof*⟩

lemma *frameChainEq′*:
fixes *xvec* :: *name list*
and Ψ :: $'a::fs\text{-}name$
and *yvec* :: *name list*
and Ψ' :: $'a::fs\text{-}name$

assumes $\langle xvec, \Psi\rangle = \langle yvec, \Psi'\rangle$
and *xvec* $\sharp*$ *yvec*
and *distinct xvec*
and *distinct yvec*

obtains *p* **where** (*set p*) $\subseteq$ (*set xvec*) $\times$ *set* ($p \cdot$ *xvec*) **and** *distinctPerm p* **and** *yvec* $= p \cdot$ *xvec* **and** $\Psi' = p \cdot \Psi$
⟨*proof*⟩

lemma *frameEq*[*simp*]:
fixes A_F :: *name list*
and Ψ :: $'a::fs\text{-}name$
and Ψ' :: $'a$

shows $\langle A_F, \Psi\rangle = \langle \varepsilon, \Psi'\rangle = (A_F = [] \wedge \Psi = \Psi')$
and $\langle \varepsilon, \Psi'\rangle = \langle A_F, \Psi\rangle = (A_F = [] \wedge \Psi = \Psi')$
⟨*proof*⟩

lemma *distinctFrame*:
fixes A_F :: *name list*
and Ψ_F :: $'a::fs\text{-}name$
and C :: $'b::fs\text{-}name$

assumes A_F $\sharp*$ C

obtains A_F' **where** $\langle A_F, \Psi_F\rangle = \langle A_F', \Psi_F\rangle$ **and** *distinct* A_F' **and** $A_F' \sharp* C$
⟨*proof*⟩

lemma *freshFrame*:
fixes F :: ('*a*::*fs-name*) *frame*
and C :: '*b* ::*fs-name*

obtains A_F Ψ_F **where** $F = \langle A_F, \Psi_F\rangle$ **and** *distinct* A_F **and** $A_F \sharp* C$
⟨*proof*⟩

locale *assertionAux* =
fixes *SCompose* :: '*b*::*fs-name* ⇒ '*b* ⇒ '*b* (**infixr** ‹⊗› *80*)
and *SImp* :: '*b* ⇒ '*c*::*fs-name* ⇒ *bool* (‹- ⊢ -› [*70*, *70*] *70*)
and *SBottom* :: '*b* (‹⊥› *90*)
and *SChanEq* :: '*a*::*fs-name* ⇒ '*a* ⇒ '*c* (‹- ↔ -› [*80*, *80*] *80*)

assumes *statEqvt*[*eqvt*]: $\bigwedge p$::*name prm*. $p \cdot (\Psi \vdash \Phi) = (p \cdot \Psi) \vdash (p \cdot \Phi)$
and *statEqvt'*[*eqvt*]: $\bigwedge p$::*name prm*. $p \cdot (\Psi \otimes \Psi') = (p \cdot \Psi) \otimes (p \cdot \Psi')$
and *statEqvt''*[*eqvt*]: $\bigwedge p$::*name prm*. $p \cdot (M \leftrightarrow N) = (p \cdot M) \leftrightarrow (p \cdot N)$
and *permBottom*[*eqvt*]: $\bigwedge p$::*name prm*. $(p \cdot SBottom) = SBottom$

begin

lemma *statClosed*:
fixes Ψ :: '*b*
and φ :: '*c*
and p :: *name prm*

assumes $\Psi \vdash \varphi$

shows $(p \cdot \Psi) \vdash (p \cdot \varphi)$
⟨*proof*⟩

lemma *compSupp*:
fixes Ψ :: '*b*
and Ψ' :: '*b*

shows (*supp*($\Psi \otimes \Psi'$)::*name set*) ⊆ ((*supp* Ψ) ∪ (*supp* Ψ'))
⟨*proof*⟩

lemma *chanEqSupp*:
fixes M :: '*a*
and N :: '*a*

shows (*supp*($M \leftrightarrow N$)::*name set*) ⊆ ((*supp* M) ∪ (*supp* N))
⟨*proof*⟩

lemma *freshComp*[*intro*]:
fixes x :: *name*

and $\Psi :: 'b$
and $\Psi' :: 'b$

assumes $x \sharp \Psi$
and $x \sharp \Psi'$

shows $x \sharp \Psi \otimes \Psi'$
⟨*proof*⟩

lemma *freshCompChain*[*intro*]:
fixes $xvec :: name\ list$
and $Xs :: name\ set$
and $\Psi :: 'b$
and $\Psi' :: 'b$

shows $[\![xvec \sharp* \Psi;\ xvec \sharp* \Psi']\!] \Longrightarrow xvec \sharp* (\Psi \otimes \Psi')$
and $[\![Xs \sharp* \Psi;\ Xs \sharp* \Psi']\!] \Longrightarrow Xs \sharp* (\Psi \otimes \Psi')$
⟨*proof*⟩

lemma *freshChanEq*[*intro*]:
fixes $x :: name$
and $M :: 'a$
and $N :: 'a$

assumes $x \sharp M$
and $x \sharp N$

shows $x \sharp M \leftrightarrow N$
⟨*proof*⟩

lemma *freshChanEqChain*[*intro*]:
fixes $xvec :: name\ list$
and $Xs :: name\ set$
and $M :: 'a$
and $N :: 'a$

shows $[\![xvec \sharp* M;\ xvec \sharp* N]\!] \Longrightarrow xvec \sharp* (M \leftrightarrow N)$
and $[\![Xs \sharp* M;\ Xs \sharp* N]\!] \Longrightarrow Xs \sharp* (M \leftrightarrow N)$
⟨*proof*⟩

lemma *suppBottom*[*simp*]:
shows $((supp\ SBottom)::name\ set) = \{\}$
⟨*proof*⟩

lemma *freshBottom*[*simp*]:
fixes $x :: name$

shows $x \sharp \bot$
⟨*proof*⟩

lemma *freshBottoChain*[*simp*]:
  fixes *xvec* :: *name list*
  and *Xs* :: *name set*

  shows *xvec* $\sharp*$ $(\bot)$
  and *Xs* $\sharp*$ $(\bot)$
⟨*proof*⟩

lemma *chanEqClosed*:
  fixes Ψ :: $'b$
  and M :: $'a$
  and N :: $'a$
  and p :: *name prm*

  assumes $\Psi \vdash M \leftrightarrow N$

  shows $(p \cdot \Psi) \vdash (p \cdot M) \leftrightarrow (p \cdot N)$
⟨*proof*⟩

definition
  AssertionStatImp :: $'b \Rightarrow 'b \Rightarrow$ *bool* (**infix** $\langle\hookrightarrow\rangle$ *70*)
  where $(\Psi \hookrightarrow \Psi') \equiv (\forall \Phi.\ \Psi \vdash \Phi \longrightarrow \Psi' \vdash \Phi)$

definition
  AssertionStatEq :: $'b \Rightarrow 'b \Rightarrow$ *bool* (**infix** $\langle\simeq\rangle$ *70*)
  where $(\Psi \simeq \Psi') \equiv \Psi \hookrightarrow \Psi' \wedge \Psi' \hookrightarrow \Psi$

lemma *statImpEnt*:
  fixes Ψ :: $'b$
  and Ψ' :: $'b$
  and Φ :: $'c$

  assumes $\Psi \hookrightarrow \Psi'$
  and $\Psi \vdash \Phi$

  shows $\Psi' \vdash \Phi$
⟨*proof*⟩

lemma *statEqEnt*:
  fixes Ψ :: $'b$
  and Ψ' :: $'b$
  and Φ :: $'c$

  assumes $\Psi \simeq \Psi'$
  and $\Psi \vdash \Phi$

  shows $\Psi' \vdash \Phi$
⟨*proof*⟩

lemma *AssertionStatImpClosed*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and p :: *name prm*

assumes $\Psi \hookrightarrow \Psi'$

shows $(p \cdot \Psi) \hookrightarrow (p \cdot \Psi')$
⟨*proof*⟩

lemma *AssertionStatEqClosed*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and p :: *name prm*

assumes $\Psi \simeq \Psi'$

shows $(p \cdot \Psi) \simeq (p \cdot \Psi')$
⟨*proof*⟩

lemma *AssertionStatImpEqvt*[*eqvt*]:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and p :: *name prm*

shows $(p \cdot (\Psi \hookrightarrow \Psi')) = ((p \cdot \Psi) \hookrightarrow (p \cdot \Psi'))$
⟨*proof*⟩

lemma *AssertionStatEqEqvt*[*eqvt*]:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and p :: *name prm*

shows $(p \cdot (\Psi \simeq \Psi')) = ((p \cdot \Psi) \simeq (p \cdot \Psi'))$
⟨*proof*⟩

lemma *AssertionStatImpRefl*[*simp*]:
fixes Ψ :: $'b$

shows $\Psi \hookrightarrow \Psi$
⟨*proof*⟩

lemma *AssertionStatEqRefl*[*simp*]:
fixes Ψ :: $'b$

shows $\Psi \simeq \Psi$
⟨*proof*⟩

lemma *AssertionStatEqSym*:
fixes Ψ :: *'b*
and Ψ' :: *'b*

assumes $\Psi \simeq \Psi'$

shows $\Psi' \simeq \Psi$
⟨*proof*⟩

lemma *AssertionStatImpTrans*:
fixes Ψ :: *'b*
and Ψ' :: *'b*
and Ψ'' :: *'b*

assumes $\Psi \hookrightarrow \Psi'$
and $\Psi' \hookrightarrow \Psi''$

shows $\Psi \hookrightarrow \Psi''$
⟨*proof*⟩

lemma *AssertionStatEqTrans*:
fixes Ψ :: *'b*
and Ψ' :: *'b*
and Ψ'' :: *'b*

assumes $\Psi \simeq \Psi'$
and $\Psi' \simeq \Psi''$

shows $\Psi \simeq \Psi''$
⟨*proof*⟩

definition
FrameImp :: *'b*::*fs-name frame* ⇒ *'c* ⇒ *bool* (**infixl** ‹$\vdash_F$› *70*)
where $(F \vdash_F \Phi) = (\exists A_F\ \Psi_F.\ F = \langle A_F, \Psi_F\rangle \wedge A_F \sharp* \Phi \wedge (\Psi_F \vdash \Phi))$

lemma *frameImpI*:
fixes F :: *'b frame*
and φ :: *'c*
and A_F :: *name list*
and Ψ_F :: *'b*

assumes $F = \langle A_F, \Psi_F\rangle$
and $A_F \sharp* \varphi$
and $\Psi_F \vdash \varphi$

shows $F \vdash_F \varphi$
⟨*proof*⟩

lemma *frameImpAlphaEnt*:

fixes A_F :: *name list*
and Ψ_F :: $'b$
and A_F' :: *name list*
and Ψ_F' :: $'b$
and φ :: $'c$

assumes $\langle A_F, \Psi_F\rangle = \langle A_F', \Psi_F'\rangle$
and $A_F \sharp* \varphi$
and $A_F' \sharp* \varphi$
and $\Psi_F' \vdash \varphi$

shows $\Psi_F \vdash \varphi$
⟨*proof*⟩

lemma *frameImpEAux*:
fixes F :: $'b$ *frame*
and Φ :: $'c$

assumes $F \vdash_F \Phi$
and $F = \langle A_F, \Psi_F\rangle$
and $A_F \sharp* \Phi$

shows $\Psi_F \vdash \Phi$
⟨*proof*⟩

lemma *frameImpE*:
fixes F :: $'b$ *frame*
and Φ :: $'c$

assumes $\langle A_F, \Psi_F\rangle \vdash_F \Phi$
and $A_F \sharp* \Phi$

shows $\Psi_F \vdash \Phi$
⟨*proof*⟩

lemma *frameImpClosed*:
fixes F :: $'b$ *frame*
and Φ :: $'c$
and p :: *name prm*

assumes $F \vdash_F \Phi$

shows $(p \cdot F) \vdash_F (p \cdot \Phi)$
⟨*proof*⟩

lemma *frameImpEqvt*[*eqvt*]:
fixes F :: $'b$ *frame*
and Φ :: $'c$
and p :: *name prm*

shows $(p \cdot (F \vdash_F \Phi)) = (p \cdot F) \vdash_F (p \cdot \Phi)$
⟨*proof*⟩

lemma *frameImpEmpty*[*simp*]:
fixes $\Psi :: 'b$
and $\varphi :: 'c$

shows $\langle \varepsilon, \Psi \rangle \vdash_F \varphi = \Psi \vdash \varphi$
⟨*proof*⟩

definition
FrameStatImp :: *'b frame* ⇒ *'b frame*⇒ *bool* (**infix** ‹$\hookrightarrow_F$› *70*)
where $(F \hookrightarrow_F G) \equiv (\forall \varphi.\ F \vdash_F \varphi \longrightarrow G \vdash_F \varphi)$

definition
FrameStatEq :: *'b frame* ⇒ *'b frame*⇒ *bool* (**infix** ‹$\simeq_F$› *70*)
where $(F \simeq_F G) \equiv F \hookrightarrow_F G \wedge G \hookrightarrow_F F$

lemma *FrameStatImpClosed*:
fixes $F :: 'b$ *frame*
and $G :: 'b$ *frame*
and $p ::$ *name prm*

assumes $F \hookrightarrow_F G$

shows $(p \cdot F) \hookrightarrow_F (p \cdot G)$
⟨*proof*⟩

lemma *FrameStatEqClosed*:
fixes $F :: 'b$ *frame*
and $G :: 'b$ *frame*
and $p ::$ *name prm*

assumes $F \simeq_F G$

shows $(p \cdot F) \simeq_F (p \cdot G)$
⟨*proof*⟩

lemma *FrameStatImpEqvt*[*eqvt*]:
fixes $F :: 'b$ *frame*
and $G :: 'b$ *frame*
and $p ::$ *name prm*

shows $(p \cdot (F \hookrightarrow_F G)) = ((p \cdot F) \hookrightarrow_F (p \cdot G))$
⟨*proof*⟩

lemma *FrameStatEqEqvt*[*eqvt*]:
fixes $F :: 'b$ *frame*

and $G :: 'b\ frame$
and $p :: name\ prm$

shows $(p \cdot (F \simeq_F G)) = ((p \cdot F) \simeq_F (p \cdot G))$
⟨*proof*⟩

lemma *FrameStatImpRefl*[*simp*]:
fixes $F :: 'b\ frame$

shows $F \hookrightarrow_F F$
⟨*proof*⟩

lemma *FrameStatEqRefl*[*simp*]:
fixes $F :: 'b\ frame$

shows $F \simeq_F F$
⟨*proof*⟩

lemma *FrameStatEqSym*:
fixes $F :: 'b\ frame$
and $G :: 'b\ frame$

assumes $F \simeq_F G$

shows $G \simeq_F F$
⟨*proof*⟩

lemma *FrameStatImpTrans*:
fixes $F :: 'b\ frame$
and $G :: 'b\ frame$
and $H :: 'b\ frame$

assumes $F \hookrightarrow_F G$
and $G \hookrightarrow_F H$

shows $F \hookrightarrow_F H$
⟨*proof*⟩

lemma *FrameStatEqTrans*:
fixes $F :: 'b\ frame$
and $G :: 'b\ frame$
and $H :: 'b\ frame$

assumes $F \simeq_F G$
and $G \simeq_F H$

shows $F \simeq_F H$
⟨*proof*⟩

lemma *fsCompose*[*simp*]: *finite*((*supp SCompose*)::*name set*)
⟨*proof*⟩

nominal-primrec
  insertAssertion :: *'b frame* ⇒ *'b* ⇒ *'b frame*
where
  insertAssertion (*FAssert* Ψ) Ψ′ = *FAssert* (Ψ′ ⊗ Ψ)
| *x* ♯ Ψ′ ⟹ *insertAssertion* ((⦇ν*x*⦈)*F*) Ψ′ = (⦇ν*x*⦈)(*insertAssertion F* Ψ′)
⟨*proof*⟩

lemma *insertAssertionEqvt*[*eqvt*]:
  fixes *p* :: *name prm*
  and *F* :: *'b frame*
  and Ψ :: *'b*

  shows *p* ∙ (*insertAssertion F* Ψ) = *insertAssertion* (*p* ∙ *F*) (*p* ∙ Ψ)
⟨*proof*⟩

nominal-primrec
  mergeFrame :: *'b frame* ⇒ *'b frame* ⇒ *'b frame*
where
  mergeFrame (*FAssert* Ψ) *G* = *insertAssertion G* Ψ
| *x* ♯ *G* ⟹ *mergeFrame* ((⦇ν*x*⦈)*F*) *G* = (⦇ν*x*⦈)(*mergeFrame F G*)
⟨*proof*⟩

notation *mergeFrame* (**infixr** ‹⊗$_F$› *80*)

abbreviation
  frameBottomJudge (‹⊥$_F$›) **where** ⊥$_F$ ≡ (*FAssert SBottom*)

lemma *mergeFrameEqvt*[*eqvt*]:
  fixes *p* :: *name prm*
  and *F* :: *'b frame*
  and *G* :: *'b frame*

  shows *p* ∙ (*mergeFrame F G*) = *mergeFrame* (*p* ∙ *F*) (*p* ∙ *G*)
⟨*proof*⟩

nominal-primrec
  extractFrame :: (*'a*, *'b*, *'c*) *psi* ⇒ *'b frame*
and *extractFrame′* :: (*'a*, *'b*, *'c*) *input* ⇒ *'b frame*
and *extractFrame″* :: (*'a*, *'b*, *'c*) *psiCase* ⇒ *'b frame*

where
  extractFrame (**0**) = ⟨ε, ⊥⟩
| *extractFrame* (*M*⦇*I*) = ⟨ε, ⊥⟩
| *extractFrame* (*M*⟨*N*⟩.*P*) = ⟨ε, ⊥⟩
| *extractFrame* (*Case C*) = ⟨ε, ⊥⟩

| *extractFrame* $(P \parallel Q) = (\mathit{extractFrame}\ P) \otimes_F (\mathit{extractFrame}\ Q)$
| *extractFrame* $((\{\!|\Psi|\!\}::('a, 'b, 'c)\ \mathit{psi})) = \langle \varepsilon, \Psi \rangle$

| *extractFrame* $((\!|\nu x|\!)P) = (\!|\nu x|\!)(\mathit{extractFrame}\ P)$
| *extractFrame* $(!P) = \langle \varepsilon, \bot \rangle$

| *extractFrame'* $((\mathit{Trm}\ M\ P)::('a::\mathit{fs\text{-}name}, 'b::\mathit{fs\text{-}name}, 'c::\mathit{fs\text{-}name})\ \mathit{input}) = \langle \varepsilon, \bot \rangle$

| *extractFrame'* $(\mathit{Bind}\ x\ I) = \langle \varepsilon, \bot \rangle$

| *extractFrame''* $(\bot_c::('a::\mathit{fs\text{-}name}, 'b::\mathit{fs\text{-}name}, 'c::\mathit{fs\text{-}name})\ \mathit{psiCase}) = \langle \varepsilon, \bot \rangle$
| *extractFrame''* $(\Box\Phi \Rightarrow P\ C) = \langle \varepsilon, \bot \rangle$
⟨*proof*⟩

lemmas *extractFrameSimps* = *extractFrame-extractFrame'-extractFrame''.simps*

lemma *extractFrameEqvt*[*eqvt*]:
  fixes $p :: \mathit{name\ prm}$
  and $P :: ('a, 'b, 'c)\ \mathit{psi}$
  and $I :: ('a, 'b, 'c)\ \mathit{input}$
  and $C :: ('a, 'b, 'c)\ \mathit{psiCase}$

  shows $p \cdot (\mathit{extractFrame}\ P) = \mathit{extractFrame}\ (p \cdot P)$
  and $p \cdot (\mathit{extractFrame'}\ I) = \mathit{extractFrame'}\ (p \cdot I)$
  and $p \cdot (\mathit{extractFrame''}\ C) = \mathit{extractFrame''}\ (p \cdot C)$
⟨*proof*⟩

lemma *insertAssertionFresh*[*intro*]:
  fixes $F :: 'b\ \mathit{frame}$
  and $\Psi :: 'b$
  and $x :: \mathit{name}$

  assumes $x \sharp F$
  and $x \sharp \Psi$

  shows $x \sharp (\mathit{insertAssertion}\ F\ \Psi)$
⟨*proof*⟩

lemma *insertAssertionFreshChain*[*intro*]:
  fixes $F :: 'b\ \mathit{frame}$
  and $\Psi :: 'b$
  and $\mathit{xvec} :: \mathit{name\ list}$
  and $\mathit{Xs} :: \mathit{name\ set}$

  shows $[\![\mathit{xvec} \sharp* F; \mathit{xvec} \sharp* \Psi]\!] \Longrightarrow \mathit{xvec} \sharp* (\mathit{insertAssertion}\ F\ \Psi)$
  and $[\![\mathit{Xs} \sharp* F; \mathit{Xs} \sharp* \Psi]\!] \Longrightarrow \mathit{Xs} \sharp* (\mathit{insertAssertion}\ F\ \Psi)$
⟨*proof*⟩

lemma *mergeFrameFresh*[*intro*]:

fixes $F :: 'b\ frame$
and $G :: 'b\ frame$
and $x :: name$

shows $[\![x \sharp F; x \sharp G]\!] \Longrightarrow x \sharp (mergeFrame\ F\ G)$
⟨*proof*⟩

lemma *mergeFrameFreshChain*[*intro*]:
fixes $F :: 'b\ frame$
and $G :: 'b\ frame$
and $xvec :: name\ list$
and $Xs :: name\ set$

shows $[\![xvec \sharp* F; xvec \sharp* G]\!] \Longrightarrow xvec \sharp* (mergeFrame\ F\ G)$
and $[\![Xs \sharp* F; Xs \sharp* G]\!] \Longrightarrow Xs \sharp* (mergeFrame\ F\ G)$
⟨*proof*⟩

lemma *extractFrameFresh*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $I :: ('a, 'b, 'c)\ input$
and $C :: ('a, 'b, 'c)\ psiCase$
and $x :: name$

shows $x \sharp P \Longrightarrow x \sharp extractFrame\ P$
and $x \sharp I \Longrightarrow x \sharp extractFrame'\ I$
and $x \sharp C \Longrightarrow x \sharp extractFrame''\ C$
⟨*proof*⟩

lemma *extractFrameFreshChain*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $I :: ('a, 'b, 'c)\ input$
and $C :: ('a, 'b, 'c)\ psiCase$
and $xvec :: name\ list$
and $Xs :: name\ set$

shows $xvec \sharp* P \Longrightarrow xvec \sharp* extractFrame\ P$
and $xvec \sharp* I \Longrightarrow xvec \sharp* extractFrame'\ I$
and $xvec \sharp* C \Longrightarrow xvec \sharp* extractFrame''\ C$
and $Xs \sharp* P \Longrightarrow Xs \sharp* extractFrame\ P$
and $Xs \sharp* I \Longrightarrow Xs \sharp* extractFrame'\ I$
and $Xs \sharp* C \Longrightarrow Xs \sharp* extractFrame''\ C$
⟨*proof*⟩

lemma *guardedFrameSupp*[*simp*]:
fixes $P :: ('a, 'b, 'c)\ psi$
and $I :: ('a, 'b, 'c)\ input$
and $C :: ('a, 'b, 'c)\ psiCase$
and $x :: name$

shows *guarded P* $\Longrightarrow$ $x \sharp$ (*extractFrame P*)
and *guarded′ I* $\Longrightarrow$ $x \sharp$ (*extractFrame′ I*)
and *guarded″ C* $\Longrightarrow$ $x \sharp$ (*extractFrame″ C*)
⟨*proof*⟩

lemma *frameResChainFresh′*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *F* :: *′b frame*

shows (*xvec* $\sharp*$ ((⌊ν*yvec*⌋)*F*)) = (∀ *x* ∈ *set xvec*. *x* ∈ *set yvec* ∨ *x* $\sharp$ *F*)
⟨*proof*⟩

lemma *frameChainFresh*[*simp*]:
fixes *xvec* :: *name list*
and Ψ :: *′b*
and *Xs* :: *name set*

shows *xvec* $\sharp*$ (*FAssert* Ψ) = *xvec* $\sharp*$ Ψ
and *Xs* $\sharp*$ (*FAssert* Ψ) = *Xs* $\sharp*$ Ψ
⟨*proof*⟩

lemma *frameResChainFresh″*[*simp*]:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *F* :: *′b frame*

assumes *xvec* $\sharp*$ *yvec*

shows *xvec* $\sharp*$ ((⌊ν*yvec*⌋)*F*) = *xvec* $\sharp*$ *F*

⟨*proof*⟩

lemma *frameResChainFresh‴*[*simp*]:
fixes *x* :: *name*
and *xvec* :: *name list*
and *F* :: *′b frame*

assumes *x* $\sharp$ *xvec*

shows *x* $\sharp$ ((⌊ν*xvec*⌋)*F*) = *x* $\sharp$ *F*
⟨*proof*⟩

lemma *FFreshBottom*[*simp*]:
fixes *xvec* :: *name list*
and *Xs* :: *name set*

shows *xvec* $\sharp*$ ($\bot_F$)

and $Xs \sharp* (\bot_F)$
⟨*proof*⟩

lemma *SFreshBottom*[*simp*]:
fixes *xvec* :: *name list*
and *Xs* :: *name set*

shows *xvec* $\sharp*$ (*SBottom*)
and *Xs* $\sharp*$ (*SBottom*)
⟨*proof*⟩

lemma *freshFrameDest*[*dest*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and *xvec* :: *name list*

assumes *xvec* $\sharp* (\langle A_F, \Psi_F\rangle)$

shows *xvec* $\sharp* A_F \Longrightarrow$ *xvec* $\sharp* \Psi_F$
and $A_F \sharp*$ *xvec* $\Longrightarrow$ *xvec* $\sharp* \Psi_F$
⟨*proof*⟩

lemma *insertAssertionSimps*[*simp*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and Ψ :: *'b*

assumes $A_F \sharp* \Psi$

shows *insertAssertion* $(\langle A_F, \Psi_F\rangle)\ \Psi = \langle A_F, \Psi \otimes \Psi_F\rangle$
⟨*proof*⟩

lemma *mergeFrameSimps*[*simp*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and Ψ :: *'b*

assumes $A_F \sharp* \Psi$

shows $(\langle A_F, \Psi_F\rangle) \otimes_F \langle \varepsilon, \Psi\rangle = \langle A_F, \Psi_F \otimes \Psi\rangle$
⟨*proof*⟩

lemma *mergeFrames*[*simp*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and A_G :: *name list*
and Ψ_G :: *'b*

assumes $A_F \sharp* A_G$

and $A_F \sharp* \Psi_G$
and $A_G \sharp* \Psi_F$

shows $(\langle A_F, \Psi_F\rangle) \otimes_F (\langle A_G, \Psi_G\rangle) = (\langle (A_F @ A_G), \Psi_F \otimes \Psi_G\rangle)$
⟨*proof*⟩

lemma *frameImpResFreshLeft*:
fixes $F :: 'b$ *frame*
and $x :: name$

assumes $x \sharp F$

shows $(\!|\nu x|\!) F \hookrightarrow_F F$
⟨*proof*⟩

lemma *frameImpResFreshRight*:
fixes $F :: 'b$ *frame*
and $x :: name$

assumes $x \sharp F$

shows $F \hookrightarrow_F (\!|\nu x|\!) F$
⟨*proof*⟩

lemma *frameResFresh*:
fixes $F :: 'b$ *frame*
and $x :: name$

assumes $x \sharp F$

shows $(\!|\nu x|\!) F \simeq_F F$
⟨*proof*⟩

lemma *frameImpResPres*:
fixes $F :: 'b$ *frame*
and $G :: 'b$ *frame*
and $x :: name$

assumes $F \hookrightarrow_F G$

shows $(\!|\nu x|\!) F \hookrightarrow_F (\!|\nu x|\!) G$
⟨*proof*⟩

lemma *frameResPres*:
fixes $F :: 'b$ *frame*
and $G :: 'b$ *frame*
and $x :: name$

assumes $F \simeq_F G$

shows $(\!|\nu x|\!)F \simeq_F (\!|\nu x|\!)G$
⟨*proof*⟩

lemma *frameImpResComm*:
fixes x :: *name*
and y :: *name*
and F :: $'b$ *frame*

shows $(\!|\nu x|\!)((\!|\nu y|\!)F) \hookrightarrow_F (\!|\nu y|\!)((\!|\nu x|\!)F)$
⟨*proof*⟩

lemma *frameResComm*:
fixes x :: *name*
and y :: *name*
and F :: $'b$ *frame*

shows $(\!|\nu x|\!)((\!|\nu y|\!)F) \simeq_F (\!|\nu y|\!)((\!|\nu x|\!)F)$
⟨*proof*⟩

lemma *frameImpResCommLeft'*:
fixes x :: *name*
and $xvec$ :: *name list*
and F :: $'b$ *frame*

shows $(\!|\nu x|\!)((\!|\nu{*}xvec|\!)F) \hookrightarrow_F (\!|\nu{*}xvec|\!)((\!|\nu x|\!)F)$
⟨*proof*⟩

lemma *frameImpResCommRight'*:
fixes x :: *name*
and $xvec$ :: *name list*
and F :: $'b$ *frame*

shows $(\!|\nu{*}xvec|\!)((\!|\nu x|\!)F) \hookrightarrow_F (\!|\nu x|\!)((\!|\nu{*}xvec|\!)F)$
⟨*proof*⟩

lemma *frameResComm'*:
fixes x :: *name*
and $xvec$ :: *name list*
and F :: $'b$ *frame*

shows $(\!|\nu x|\!)((\!|\nu{*}xvec|\!)F) \simeq_F (\!|\nu{*}xvec|\!)((\!|\nu x|\!)F)$
⟨*proof*⟩

lemma *frameImpChainComm*:
fixes $xvec$ :: *name list*
and $yvec$ :: *name list*
and F :: $'b$ *frame*

shows $(\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)F) \hookrightarrow_F (\!|\nu{*}yvec|\!)((\!|\nu{*}xvec|\!)F)$
⟨*proof*⟩

lemma *frameResChainComm*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *F* :: *'b frame*

shows $(\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)F) \simeq_F (\!|\nu{*}yvec|\!)((\!|\nu{*}xvec|\!)F)$
⟨*proof*⟩

lemma *frameImpNilStatEq*[*simp*]:
fixes Ψ :: *'b*
and Ψ' :: *'b*

shows $(\langle\varepsilon, \Psi\rangle \hookrightarrow_F \langle\varepsilon, \Psi'\rangle) = (\Psi \hookrightarrow \Psi')$
⟨*proof*⟩

lemma *frameNilStatEq*[*simp*]:
fixes Ψ :: *'b*
and Ψ' :: *'b*

shows $(\langle\varepsilon, \Psi\rangle \simeq_F \langle\varepsilon, \Psi'\rangle) = (\Psi \simeq \Psi')$
⟨*proof*⟩

lemma *extractFrameChainStatImp*:
fixes *xvec* :: *name list*
and *P* :: *('a, 'b, 'c) psi*

shows *extractFrame*$((\!|\nu{*}xvec|\!)P) \hookrightarrow_F (\!|\nu{*}xvec|\!)(extractFrame\ P)$
⟨*proof*⟩

lemma *extractFrameChainStatEq*:
fixes *xvec* :: *name list*
and *P* :: *('a, 'b, 'c) psi*

shows *extractFrame*$((\!|\nu{*}xvec|\!)P) \simeq_F (\!|\nu{*}xvec|\!)(extractFrame\ P)$
⟨*proof*⟩

lemma *insertAssertionExtractFrameFreshImp*:
fixes *xvec* :: *name list*
and Ψ :: *'b*
and *P* :: *('a, 'b, 'c) psi*

assumes *xvec* $\sharp*$ Ψ

shows *insertAssertion*$(extractFrame((\!|\nu{*}xvec|\!)P))\ \Psi \hookrightarrow_F (\!|\nu{*}xvec|\!)(insertAssertion\ (extractFrame\ P)\ \Psi)$

⟨*proof*⟩

lemma *insertAssertionExtractFrameFresh*:
fixes *xvec* :: *name list*
and Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*

assumes *xvec* ♯* Ψ

shows *insertAssertion*(*extractFrame*((⌊ν*xvec⌋)*P*)) Ψ ≃$_F$ (⌊ν*xvec⌋)(*insertAssertion* (*extractFrame P*) Ψ)
⟨*proof*⟩

lemma *frameImpResChainPres*:
fixes *F* :: *'b frame*
and *G* :: *'b frame*
and *xvec* :: *name list*

assumes *F* ↪$_F$ *G*

shows (⌊ν*xvec⌋)*F* ↪$_F$ (⌊ν*xvec⌋)*G*
⟨*proof*⟩

lemma *frameResChainPres*:
fixes *F* :: *'b frame*
and *G* :: *'b frame*
and *xvec* :: *name list*

assumes *F* ≃$_F$ *G*

shows (⌊ν*xvec⌋)*F* ≃$_F$ (⌊ν*xvec⌋)*G*
⟨*proof*⟩

lemma *insertAssertionE*:
fixes *F* :: (*'b*::*fs-name*) *frame*
and Ψ :: *'b*
and Ψ′ :: *'b*
and A_F :: *name list*

assumes *insertAssertion F* Ψ = ⟨A_F, Ψ′⟩
and A_F ♯* *F*
and A_F ♯* Ψ
and *distinct* A_F

obtains Ψ_F **where** *F* = ⟨A_F, Ψ_F⟩ **and** Ψ′ = Ψ ⊗ Ψ_F
⟨*proof*⟩

lemma *mergeFrameE*:
fixes *F* :: *'b frame*

and G :: *'b frame*
and A_{FG} :: *name list*
and Ψ_{FG} :: *'b*

assumes *mergeFrame* F $G = \langle A_{FG}, \Psi_{FG}\rangle$
and *distinct* A_{FG}
and $A_{FG} \sharp* F$
and $A_{FG} \sharp* G$

obtains A_F Ψ_F A_G Ψ_G **where** $A_{FG} = A_F@A_G$ **and** $\Psi_{FG} = \Psi_F \otimes \Psi_G$ **and** $F = \langle A_F, \Psi_F\rangle$ **and** $G = \langle A_G, \Psi_G\rangle$ **and** $A_F \sharp* \Psi_G$ **and** $A_G \sharp* \Psi_F$
⟨*proof*⟩

lemma *mergeFrameRes1*[*simp*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and x :: *name*
and A_G :: *name list*
and Ψ_G :: *'b*

assumes $A_F \sharp* \Psi_G$
and $A_F \sharp* A_G$
and $x \sharp A_F$
and $x \sharp \Psi_F$
and $A_G \sharp* \Psi_F$

shows $(\langle A_F, \Psi_F\rangle) \otimes_F (\langle\!|\nu x|\!\rangle(\langle A_G, \Psi_G\rangle)) = (\langle(A_F@x\#A_G), \Psi_F \otimes \Psi_G\rangle)$
⟨*proof*⟩

lemma *mergeFrameRes2*[*simp*]:
fixes A_F :: *name list*
and Ψ_F :: *'b*
and x :: *name*
and A_G :: *name list*
and Ψ_G :: *'b*

assumes $A_F \sharp* \Psi_G$
and $A_G \sharp* A_F$
and $x \sharp A_F$
and $x \sharp \Psi_F$
and $A_G \sharp* \Psi_F$

shows $(\langle A_F, \Psi_F\rangle) \otimes_F (\langle\!|\nu x|\!\rangle(\langle A_G, \Psi_G\rangle)) = (\langle(A_F@x\#A_G), \Psi_F \otimes \Psi_G\rangle)$
⟨*proof*⟩

lemma *insertAssertionResChain*[*simp*]:
fixes *xvec* :: *name list*
and F :: *'b frame*
and Ψ :: *'b*

assumes *xvec* $\sharp*$ Ψ

shows *insertAssertion* $((\!|\nu*xvec|\!)F)$ Ψ $=$ $(\!|\nu*xvec|\!)(insertAssertion\ F\ \Psi)$
⟨*proof*⟩

lemma *extractFrameResChain*[*simp*]:
fixes *xvec* :: *name list*
and *P* :: ('*a*, '*b*, '*c*) *psi*

shows *extractFrame*$((\!|\nu*xvec|\!)P)$ $=$ $(\!|\nu*xvec|\!)(extractFrame\ P)$
⟨*proof*⟩

lemma *frameResFreshChain*:
fixes *xvec* :: *name list*
and *F* :: '*b frame*

assumes *xvec* $\sharp*$ *F*

shows $(\!|\nu*xvec|\!)F \simeq_F F$
⟨*proof*⟩

end

locale *assertion* = *assertionAux SCompose SImp SBottom SChanEq*
for *SCompose* :: '*b*::*fs-name* ⇒ '*b* ⇒ '*b*
and *SImp* :: '*b* ⇒ '*c*::*fs-name* ⇒ *bool*
and *SBottom* :: '*b*
and *SChanEq* :: '*a*::*fs-name* ⇒ '*a* ⇒ '*c* +

assumes *chanEqSym*: *SImp* Ψ (*SChanEq M N*) $\Longrightarrow$ *SImp* Ψ (*SChanEq N M*)
and *chanEqTrans*: $[\![$*SImp* Ψ (*SChanEq M N*); *SImp* Ψ (*SChanEq N L*)$]\!] \Longrightarrow$ *SImp* Ψ (*SChanEq M L*)
and *Composition*: *assertionAux.AssertionStatEq SImp* Ψ Ψ' $\Longrightarrow$ *assertionAux.AssertionStatEq SImp* (*SCompose* Ψ Ψ'') (*SCompose* Ψ' Ψ'')
and *Identity*: *assertionAux.AssertionStatEq SImp* (*SCompose* Ψ *SBottom*) Ψ
and *Associativity*: *assertionAux.AssertionStatEq SImp* (*SCompose* (*SCompose* Ψ Ψ') Ψ'') (*SCompose* Ψ (*SCompose* Ψ' Ψ''))
and *Commutativity*: *assertionAux.AssertionStatEq SImp* (*SCompose* Ψ Ψ') (*SCompose* Ψ' Ψ)

begin

notation *SCompose* (**infixr** ‹⊗› *90*)
notation *SImp* (‹- ⊢ -› [*85*, *85*] *85*)
notation *SChanEq* (‹- ↔ -› [*90*, *90*] *90*)
notation *SBottom* (‹⊥› *90*)

lemma *compositionSym*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and Ψ'' :: $'b$

assumes $\Psi \simeq \Psi'$

shows $\Psi'' \otimes \Psi \simeq \Psi'' \otimes \Psi'$
⟨*proof*⟩

lemma *Composition'*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and Ψ'' :: $'b$
and Ψ''' :: $'b$

assumes $\Psi \simeq \Psi'$
and $\Psi'' \simeq \Psi'''$

shows $\Psi \otimes \Psi'' \simeq \Psi' \otimes \Psi'''$
⟨*proof*⟩

lemma *composition'*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and Ψ'' :: $'b$
and Ψ''' :: $'b$

assumes $\Psi \simeq \Psi'$

shows $(\Psi \otimes \Psi'') \otimes \Psi''' \simeq (\Psi' \otimes \Psi'') \otimes \Psi'''$
⟨*proof*⟩

lemma *associativitySym*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and Ψ'' :: $'b$

shows $(\Psi \otimes \Psi') \otimes \Psi'' \simeq (\Psi \otimes \Psi'') \otimes \Psi'$
⟨*proof*⟩

lemma *frameIntAssociativity*:
fixes A_F :: *name list*
and Ψ :: $'b$
and Ψ' :: $'b$
and Ψ'' :: $'b$

shows $\langle A_F, (\Psi \otimes \Psi') \otimes \Psi'' \rangle \simeq_F \langle A_F, \Psi \otimes (\Psi' \otimes \Psi'') \rangle$

⟨*proof*⟩

lemma *frameIntCommutativity*:
fixes A_F :: *name list*
and Ψ :: *'b*
and Ψ' :: *'b*

shows $\langle A_F, \Psi \otimes \Psi' \rangle \simeq_F \langle A_F, \Psi' \otimes \Psi \rangle$
⟨*proof*⟩

lemma *frameIntIdentity*:
fixes A_F :: *name list*
and Ψ_F :: *'b*

shows $\langle A_F, \Psi_F \otimes SBottom \rangle \simeq_F \langle A_F, \Psi_F \rangle$
⟨*proof*⟩

lemma *frameIntComposition*:
fixes Ψ :: *'b*
and Ψ' :: *'b*
and A_F :: *name list*
and Ψ_F :: *'b*

assumes $\Psi \simeq \Psi'$

shows $\langle A_F, \Psi \otimes \Psi_F \rangle \simeq_F \langle A_F, \Psi' \otimes \Psi_F \rangle$
⟨*proof*⟩

lemma *frameIntCompositionSym*:
fixes Ψ :: *'b*
and Ψ' :: *'b*
and A_F :: *name list*
and Ψ_F :: *'b*

assumes $\Psi \simeq \Psi'$

shows $\langle A_F, \Psi_F \otimes \Psi \rangle \simeq_F \langle A_F, \Psi_F \otimes \Psi' \rangle$
⟨*proof*⟩

lemma *frameCommutativity*:
fixes F :: *'b frame*
and G :: *'b frame*

shows $F \otimes_F G \simeq_F G \otimes_F F$
⟨*proof*⟩

lemma *frameScopeExt*:
fixes x :: *name*
and F :: *'b frame*

and $G :: 'b\ frame$

assumes $x \sharp F$

shows $(\!|\nu x|\!)(F \otimes_F G) \simeq_F F \otimes_F ((\!|\nu x|\!) G)$
⟨*proof*⟩

lemma *insertDoubleAssertionStatEq*:
fixes $F :: 'b\ frame$
and $\Psi :: 'b$
and $\Psi' :: 'b$

shows $insertAssertion(insertAssertion\ F\ \Psi)\ \Psi' \simeq_F (insertAssertion\ F)\ (\Psi \otimes \Psi')$
⟨*proof*⟩

lemma *guardedStatEq*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $I :: ('a, 'b, 'c)\ input$
and $C :: ('a, 'b, 'c)\ psiCase$
and $A_P :: name\ list$
and $\Psi_P :: 'b$

shows $[\![guarded\ P;\ extractFrame\ P = \langle A_P, \Psi_P\rangle]\!] \Longrightarrow \Psi_P \simeq \bot \wedge supp\ \Psi_P = (\{\}::name\ set)$
and $[\![guarded'\ I;\ extractFrame'\ I = \langle A_P, \Psi_P\rangle]\!] \Longrightarrow \Psi_P \simeq \bot \wedge supp\ \Psi_P = (\{\}::name\ set)$
and $[\![guarded''\ C;\ extractFrame''\ C = \langle A_P, \Psi_P\rangle]\!] \Longrightarrow \Psi_P \simeq \bot \wedge supp\ \Psi_P = (\{\}::name\ set)$
⟨*proof*⟩

end

end

theory *Semantics*
imports *Frame*
begin

nominal-datatype $('a, 'b, 'c)\ boundOutput =$
BOut $'a::fs\text{-}name\ ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$ (‹- ≺'' -› [110, 110] 110)
| *BStep* «*name*» $('a, 'b, 'c)\ boundOutput$ (‹(\|ν-\|)-› [110, 110] 110)

primrec $BOresChain :: name\ list \Rightarrow ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput \Rightarrow$
$('a, 'b, 'c)\ boundOutput$ **where**
Base: $BOresChain\ []\ B = B$
| *Step*: $BOresChain\ (x\#xs)\ B = (\!|\nu x|\!)(BOresChain\ xs\ B)$

abbreviation
BOresChainJudge ($\langle\!\langle(\!|\nu*\text{-}|\!)\text{-}\rangle\!\rangle$ [80, 80] 80) **where** $(\!|\nu*xvec|\!)B \equiv BOresChain\ xvec\ B$

lemma *BOresChainEqvt*[*eqvt*]:
fixes *perm* :: *name prm*
and *lst* :: *name list*
and *B* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *boundOutput*

shows $perm \cdot ((\!|\nu*xvec|\!)B) = (\!|\nu*(perm \cdot xvec)|\!)(perm \cdot B)$
⟨*proof*⟩

lemma *BOresChainSimps*[*simp*]:
fixes *xvec* :: *name list*
and *N* :: '*a*::*fs-name*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *N'* :: '*a*
and *P'* :: ('*a*, '*b*, '*c*) *psi*
and *B* :: ('*a*, '*b*, '*c*) *boundOutput*
and *B'* :: ('*a*, '*b*, '*c*) *boundOutput*

shows $((\!|\nu*xvec|\!)N \prec' P = N' \prec' P') = (xvec = [] \wedge N = N' \wedge P = P')$
and $(N' \prec' P' = (\!|\nu*xvec|\!)N \prec' P) = (xvec = [] \wedge N = N' \wedge P = P')$
and $(N' \prec' P' = N \prec' P) = (N = N' \wedge P = P')$
and $((\!|\nu*xvec|\!)B = (\!|\nu*xvec|\!)B') = (B = B')$
⟨*proof*⟩

lemma *outputFresh*[*simp*]:
fixes *Xs* :: *name set*
and *xvec* :: *name list*
and *N* :: '*a*::*fs-name*
and *P* :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*

shows $(Xs \sharp* (N \prec' P)) = ((Xs \sharp* N) \wedge (Xs \sharp* P))$
and $(xvec \sharp* (N \prec' P)) = ((xvec \sharp* N) \wedge (xvec \sharp* P))$
⟨*proof*⟩

lemma *boundOutputFresh*:
fixes *x* :: *name*
and *xvec* :: *name list*
and *B* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *boundOutput*

shows $(x \sharp ((\!|\nu*xvec|\!)B)) = (x \in set\ xvec \vee x \sharp B)$
⟨*proof*⟩

lemma *boundOutputFreshSet*:
fixes *Xs* :: *name set*
and *xvec* :: *name list*
and *B* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *boundOutput*

and $yvec :: name\ list$
and $x :: name$

shows $Xs \sharp* (\langle\!|\nu* xvec|\!\rangle B) = (\forall x \in Xs.\ x \in set\ xvec \vee x \sharp B)$
and $yvec \sharp* (\langle\!|\nu* xvec|\!\rangle B) = (\forall x \in (set\ yvec).\ x \in set\ xvec \vee x \sharp B)$
and $Xs \sharp* (\langle\!|\nu x|\!\rangle B) = Xs \sharp* [x].B$
and $xvec \sharp* (\langle\!|\nu x|\!\rangle B) = xvec \sharp* [x].B$
⟨*proof*⟩

lemma *BOresChainSupp*:
fixes $xvec :: name\ list$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$

shows $(supp(\langle\!|\nu* xvec|\!\rangle B)::name\ set) = (supp\ B) - (supp\ xvec)$
⟨*proof*⟩

lemma *boundOutputFreshSimps*[*simp*]:
fixes $Xs :: name\ set$
and $xvec :: name\ list$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
and $yvec :: name\ list$
and $x :: name$

shows $Xs \sharp* xvec \Longrightarrow (Xs \sharp* (\langle\!|\nu* xvec|\!\rangle B)) = (Xs \sharp* B)$
and $yvec \sharp* xvec \Longrightarrow yvec \sharp* (\langle\!|\nu* xvec|\!\rangle B) = yvec \sharp* B$
and $xvec \sharp* (\langle\!|\nu* xvec|\!\rangle B)$
and $x \sharp xvec \Longrightarrow x \sharp \langle\!|\nu* xvec|\!\rangle B = x \sharp B$
⟨*proof*⟩

lemma *boundOutputChainAlpha*:
fixes $p :: name\ prm$
and $xvec :: name\ list$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
and $yvec :: name\ list$

assumes *xvecFreshB*: $(p \cdot xvec) \sharp* B$
and S: $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$
and $(set\ xvec) \subseteq (set\ yvec)$

shows $(\langle\!|\nu* yvec|\!\rangle B) = (\langle\!|\nu* (p \cdot yvec)|\!\rangle (p \cdot B))$
⟨*proof*⟩

lemma *boundOutputChainAlpha'*:
fixes $p :: name\ prm$
and $xvec :: name\ list$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
and $yvec :: name\ list$
and $zvec :: name\ list$

assumes *xvecFreshB*: $xvec \sharp* B$
and S: $set\ p \subseteq set\ xvec \times set\ yvec$
and $yvec \sharp* (\langle\!|\nu* zvec|\!\rangle B)$

shows $(\langle\!|\nu* zvec|\!\rangle B) = (\langle\!|\nu*(p \cdot zvec)|\!\rangle(p \cdot B))$
⟨*proof*⟩

lemma *boundOutputChainAlpha''*:
fixes p :: *name prm*
and $xvec$ :: *name list*
and M :: *'a::fs-name*
and P :: *('a::fs-name, 'b::fs-name, 'c::fs-name) psi*
and $yvec$ :: *name list*

assumes $(p \cdot xvec) \sharp* M$
and $(p \cdot xvec) \sharp* P$
and $set\ p \subseteq set\ xvec \times set\ (p \cdot xvec)$
and $(set\ xvec) \subseteq (set\ yvec)$

shows $(\langle\!|\nu* yvec|\!\rangle M \prec' P) = (\langle\!|\nu*(p \cdot yvec)|\!\rangle(p \cdot M) \prec' (p \cdot P))$
⟨*proof*⟩

lemma *boundOutputChainSwap*:
fixes x :: *name*
and y :: *name*
and N :: *'a::fs-name*
and P :: *('a, 'b::fs-name, 'c::fs-name) psi*
and $xvec$ :: *name list*

assumes $y \sharp N$
and $y \sharp P$
and $x \in (set\ xvec)$

shows $\langle\!|\nu* xvec|\!\rangle N \prec' P = \langle\!|\nu*([(x,\ y)] \cdot xvec)|\!\rangle([(x\ ,y)] \cdot N) \prec' ([(x,\ y)] \cdot P)$
⟨*proof*⟩

lemma *alphaBoundOutput*:
fixes x :: *name*
and y :: *name*
and B :: *('a::fs-name, 'b::fs-name, 'c::fs-name) boundOutput*

assumes $y \sharp B$

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

lemma *boundOutputEqFresh*:
fixes B :: *('a::fs-name, 'b::fs-name, 'c::fs-name) boundOutput*
and C :: *('a, 'b, 'c) boundOutput*

and $x :: name$
and $y :: name$

assumes $(\!|\nu x|\!)B = (\!|\nu y|\!)C$
and $x \sharp B$

shows $y \sharp C$
⟨*proof*⟩

lemma *boundOutputEqSupp*:
fixes $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
and $C :: ('a, 'b, 'c)\ boundOutput$
and $x :: name$
and $y :: name$

assumes $(\!|\nu x|\!)B = (\!|\nu y|\!)C$
and $x \in supp\ B$

shows $y \in supp\ C$
⟨*proof*⟩

lemma *boundOutputChainEq*:
fixes $xvec :: name\ list$
and $B :: ('a::fs\text{-}name, 'b::fs\text{-}name, 'c::fs\text{-}name)\ boundOutput$
and $yvec :: name\ list$
and $B' :: ('a, 'b, 'c)\ boundOutput$

assumes $(\!|\nu * xvec|\!)B = (\!|\nu * yvec|\!)B'$
and $xvec \sharp* yvec$
and $length\ xvec = length\ yvec$

shows $\exists p.\ (set\ p) \subseteq (set\ xvec) \times set\ (yvec) \wedge distinctPerm\ p \wedge\ B = p \cdot B' \wedge (set\ (map\ fst\ p)) \subseteq (supp\ B) \wedge xvec \sharp* B' \wedge yvec \sharp* B$
⟨*proof*⟩

lemma *boundOutputChainEqLength*:
fixes $xvec :: name\ list$
and $M :: 'a::fs\text{-}name$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $yvec :: name\ list$
and $N :: 'a::fs\text{-}name$
and $Q :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $(\!|\nu * xvec|\!)M \prec' P = (\!|\nu * yvec|\!)N \prec' Q$

shows $length\ xvec = length\ yvec$
⟨*proof*⟩

lemma *boundOutputChainEq'*:

fixes *xvec* :: *name list*
and *M* :: *'a::fs-name*
and *P* :: (*'a*, *'b::fs-name*, *'c::fs-name*) *psi*
and *yvec* :: *name list*
and *N* :: *'a*
and *Q* :: (*'a::fs-name*, *'b::fs-name*, *'c::fs-name*) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' Q$
and *xvec* $\sharp*$ *yvec*

shows $\exists p.\ (set\ p) \subseteq (set\ xvec) \times set\ (yvec) \wedge distinctPerm\ p \wedge\ M = p \cdot N \wedge P = p \cdot Q \wedge xvec \sharp* N \wedge xvec \sharp* Q \wedge yvec \sharp* M \wedge yvec \sharp* P$
⟨*proof*⟩

lemma *boundOutputChainEq''*:
fixes *xvec* :: *name list*
and *M* :: *'a::fs-name*
and *P* :: (*'a*, *'b::fs-name*, *'c::fs-name*) *psi*
and *yvec* :: *name list*
and *N* :: *'a*
and *Q* :: (*'a::fs-name*, *'b::fs-name*, *'c::fs-name*) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' Q$
and *xvec* $\sharp*$ *yvec*
and *distinct xvec*
and *distinct yvec*

obtains *p* **where** $(set\ p) \subseteq (set\ xvec) \times set\ (p \cdot xvec)$ **and** *distinctPerm p* **and** $yvec = p \cdot xvec$ **and** $N = p \cdot M$ **and** $Q = p \cdot P$ **and** *xvec* $\sharp*$ *N* **and** *xvec* $\sharp*$ *Q* **and** $(p \cdot xvec) \sharp* M$ **and** $(p \cdot xvec) \sharp* P$
⟨*proof*⟩

lemma *boundOutputEqSupp'*:
fixes *x* :: *name*
and *xvec* :: *name list*
and *M* :: *'a::fs-name*
and *P* :: (*'a*, *'b::fs-name*, *'c::fs-name*) *psi*
and *y* :: *name*
and *yvec* :: *name list*
and *N* :: *'a*
and *Q* :: (*'a*, *'b*, *'c*) *psi*

assumes *Eq*: $(\!|\nu x|\!)((\!|\nu{*}xvec|\!)M \prec' P) = (\!|\nu y|\!)((\!|\nu{*}yvec|\!)N \prec' Q)$
and $x \neq y$
and $x \sharp yvec$
and $x \sharp xvec$
and $y \sharp xvec$
and $y \sharp yvec$
and *xvec* $\sharp*$ *yvec*

and $x \in supp\ M$

shows $y \in supp\ N$
⟨*proof*⟩

lemma *boundOutputChainOpenIH*:
fixes *xvec* :: *name list*
and x :: *name*
and B :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *boundOutput*
and *yvec* :: *name list*
and y :: *name*
and B' :: ('*a*, '*b*, '*c*) *boundOutput*

assumes *Eq*: $(\!|\nu{*}xvec|\!)((\!|\nu x|\!)B) = (\!|\nu{*}yvec|\!)((\!|\nu y|\!)B')$
and *L*: *length xvec* = *length yvec*
and *xFreshB'*: $x \sharp B'$
and *xFreshxvec*: $x \sharp xvec$
and *xFreshyvec*: $x \sharp yvec$

shows $(\!|\nu{*}xvec|\!)B = (\!|\nu{*}yvec|\!)([(x, y)] \cdot B')$
⟨*proof*⟩

lemma *boundOutputPar1Dest*:
fixes *xvec* :: *name list*
and M :: '*a*::*fs-name*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *yvec* :: *name list*
and N :: '*a*
and Q :: ('*a*, '*b*, '*c*) *psi*
and R :: ('*a*, '*b*, '*c*) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$
and $xvec \sharp* R$
and $yvec \sharp* R$

obtains T **where** $P = T \parallel R$ **and** $(\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' Q$
⟨*proof*⟩

lemma *boundOutputPar1Dest'*:
fixes *xvec* :: *name list*
and M :: '*a*::*fs-name*
and P :: ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi*
and *yvec* :: *name list*
and N :: '*a*
and Q :: ('*a*, '*b*, '*c*) *psi*
and R :: ('*a*, '*b*, '*c*) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$
and $xvec \sharp* yvec$

obtains T p **where** $set\ p \subseteq set\ xvec \times set\ yvec$ **and** $P = T \parallel (p \cdot R)$ **and** $(\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' Q$
⟨*proof*⟩

lemma *boundOutputPar2Dest*:
fixes *xvec* :: *name list*
and M :: $'a$::*fs-name*
and P :: ($'a$, $'b$::*fs-name*, $'c$::*fs-name*) *psi*
and *yvec* :: *name list*
and N :: $'a$
and Q :: ($'a$, $'b$, $'c$) *psi*
and R :: ($'a$, $'b$, $'c$) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$
and $xvec \sharp* Q$
and $yvec \sharp* Q$

obtains T **where** $P = Q \parallel T$ **and** $(\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' R$
⟨*proof*⟩

lemma *boundOutputPar2Dest'*:
fixes *xvec* :: *name list*
and M :: $'a$::*fs-name*
and P :: ($'a$, $'b$::*fs-name*, $'c$::*fs-name*) *psi*
and *yvec* :: *name list*
and N :: $'a$
and Q :: ($'a$, $'b$, $'c$) *psi*
and R :: ($'a$, $'b$, $'c$) *psi*

assumes $(\!|\nu{*}xvec|\!)M \prec' P = (\!|\nu{*}yvec|\!)N \prec' (Q \parallel R)$
and $xvec \sharp* yvec$

obtains T p **where** $set\ p \subseteq set\ xvec \times set\ yvec$ **and** $P = (p \cdot Q) \parallel T$ **and** $(\!|\nu{*}xvec|\!)M \prec' T = (\!|\nu{*}yvec|\!)N \prec' R$
⟨*proof*⟩

lemma *boundOutputApp*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and B :: ($'a$::*fs-name*, $'b$::*fs-name*, $'c$::*fs-name*) *boundOutput*

shows $(\!|\nu{*}(xvec@yvec)|\!)B = (\!|\nu{*}xvec|\!)((\!|\nu{*}yvec|\!)B)$
⟨*proof*⟩

lemma *openInjectAuxAuxAux*:
fixes x :: *name*
and *xvec* :: *name list*

shows $\exists y\ yvec.\ x \# xvec = yvec\ @\ [y] \wedge length\ xvec = length\ yvec$
⟨*proof*⟩

lemma *openInjectAuxAux*:
fixes *xvec1* :: *name list*
and *xvec2* :: *name list*
and *yvec* :: *name list*

assumes $length(xvec1@xvec2) = length\ yvec$

shows $\exists yvec1\ yvec2.\ yvec = yvec1@yvec2 \wedge length\ xvec1 = length\ yvec1 \wedge length\ xvec2 = length\ yvec2$
⟨*proof*⟩

lemma *openInjectAux*:
fixes *xvec1* :: *name list*
and *x* :: *name*
and *xvec2* :: *name list*
and *yvec* :: *name list*

assumes $length(xvec1@x\#xvec2) = length\ yvec$

shows $\exists yvec1\ y\ yvec2.\ yvec = yvec1@y\#yvec2 \wedge length\ xvec1 = length\ yvec1 \wedge length\ xvec2 = length\ yvec2$
⟨*proof*⟩

lemma *boundOutputOpenDest*:
fixes *yvec* :: *name list*
and *M* :: $'a$::*fs-name*
and *P* :: ($'a$, $'b$::*fs-name*, $'c$::*fs-name*) *psi*
and *xvec1* :: *name list*
and *x* :: *name*
and *xvec2* :: *name list*
and *N* :: $'a$
and *Q* :: ($'a$, $'b$, $'c$) *psi*

assumes *Eq*: $(\!|\nu*(xvec1@x\#xvec2)|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$
and $x \sharp xvec1$
and $x \sharp yvec$
and $x \sharp N$
and $x \sharp Q$
and *distinct yvec*

obtains *yvec1 y yvec2* **where** *yvec*=*yvec1*@*y*#*yvec2* **and** *length xvec1* = *length yvec1* **and** *length xvec2* = *length yvec2*
and $(\!|\nu*(xvec1@xvec2)|\!)M \prec' P = (\!|\nu*(yvec1@yvec2)|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$
⟨*proof*⟩

lemma *boundOutputOpenDest'*:
fixes *yvec* :: *name list*
and *M* :: *'a::fs-name*
and *P* :: *('a, 'b::fs-name, 'c::fs-name) psi*
and *xvec1* :: *name list*
and *x* :: *name*
and *xvec2* :: *name list*
and *N* :: *'a*
and *Q* :: *('a, 'b, 'c) psi*

assumes *Eq*: $(\!|\nu*(xvec1 @ x\#xvec2)|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' Q$
and $x \sharp xvec1$
and $x \sharp yvec$
and $x \sharp N$
and $x \sharp Q$

obtains *yvec1 y yvec2* **where** *yvec=yvec1@y#yvec2* **and** *length xvec1 = length yvec1* **and** *length xvec2 = length yvec2*
and $(\!|\nu*(xvec1 @ xvec2)|\!)M \prec' P = (\!|\nu*(yvec1 @ [(x, y)] \cdot yvec2)|\!)([(x, y)] \cdot N) \prec' ([(x, y)] \cdot Q)$
⟨*proof*⟩

lemma *boundOutputScopeDest*:
fixes *xvec* :: *name list*
and *M* :: *'a::fs-name*
and *P* :: *('a, 'b::fs-name, 'c::fs-name) psi*
and *yvec* :: *name list*
and *N* :: *'a*
and *x* :: *name*
and *Q* :: *('a, 'b, 'c) psi*

assumes $(\!|\nu*xvec|\!)M \prec' P = (\!|\nu*yvec|\!)N \prec' (\!|\nu z|\!)Q$
and $z \sharp xvec$
and $z \sharp yvec$

obtains *R* **where** $P = (\!|\nu z|\!)R$ **and** $(\!|\nu*xvec|\!)M \prec' R = (\!|\nu*yvec|\!)N \prec' Q$
⟨*proof*⟩

nominal-datatype *('a, 'b, 'c) residual* =
RIn 'a::fs-name 'a ('a, 'b::fs-name, 'c::fs-name) psi
| *ROut 'a ('a, 'b, 'c) boundOutput*
| *RTau ('a, 'b, 'c) psi*

nominal-datatype *'a action* = *In 'a::fs-name 'a* (‹-(|-|)› [*90*, *90*] *90*)
| *Out 'a::fs-name name list 'a* (‹-(|ν*-|)⟨-⟩› [*90*, *90*, *90*] *90*)
| *Tau* (‹τ› *90*)

nominal-primrec *bn* :: ('*a*::*fs-name*) *action* ⇒ *name list*
  where
  bn (*M*(⌊*N*⌋)) = []
| *bn* (*M*(⌊*ν*∗*xvec*⌋⟨*N*⟩) = *xvec*
| *bn* (*τ*) = []
⟨*proof*⟩

lemma *bnEqvt*[*eqvt*]:
  fixes *p* :: *name prm*
  and *α* :: ('*a*::*fs-name*) *action*

  shows (*p* ∙ *bn α*) = *bn*(*p* ∙ *α*)
⟨*proof*⟩

nominal-primrec *create-residual* :: ('*a*::*fs-name*) *action* ⇒ ('*a*, '*b*::*fs-name*, '*c*::*fs-name*) *psi* ⇒ ('*a*, '*b*, '*c*) *residual* (‹- ≺ -› [*80*, *80*] *80*)
where
  (*M*(⌊*N*⌋)) ≺ *P* = *RIn M N P*
| *M*(⌊*ν*∗*xvec*⌋⟨*N*⟩ ≺ *P* = *ROut M* ((⌊*ν*∗*xvec*⌋(*N* ≺′ *P*))
| *τ* ≺ *P* = (*RTau P*)
⟨*proof*⟩

nominal-primrec *subject* :: ('*a*::*fs-name*) *action* ⇒ '*a option*
  where
  subject (*M*(⌊*N*⌋)) = *Some M*
| *subject* (*M*(⌊*ν*∗*xvec*⌋⟨*N*⟩) = *Some M*
| *subject* (*τ*) = *None*
⟨*proof*⟩

nominal-primrec *object* :: ('*a*::*fs-name*) *action* ⇒ '*a option*
  where
  object (*M*(⌊*N*⌋)) = *Some N*
| *object* (*M*(⌊*ν*∗*xvec*⌋⟨*N*⟩) = *Some N*
| *object* (*τ*) = *None*
⟨*proof*⟩

lemma *optionFreshChain*[*simp*]:
  fixes *xvec* :: *name list*
  and *X* :: *name set*

  shows *xvec* ♯∗ (*Some x*) = *xvec* ♯∗ *x*
  and *X* ♯∗ (*Some x*) = *X* ♯∗ *x*
  and *xvec* ♯∗ *None*
  and *X* ♯∗ *None*
⟨*proof*⟩

lemmas [*simp*] = *fresh-some fresh-none*

lemma *actionFresh*[*simp*]:

fixes $x :: name$
and $\alpha :: ('a::fs\text{-}name)\ action$

shows $(x \sharp \alpha) = (x \sharp (subject\ \alpha) \wedge x \sharp (bn\ \alpha) \wedge x \sharp (object\ \alpha))$
⟨*proof*⟩

lemma *actionFreshChain*[*simp*]:
fixes $X :: name\ set$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $xvec :: name\ list$

shows $(X \sharp* \alpha) = (X \sharp* (subject\ \alpha) \wedge X \sharp* (bn\ \alpha) \wedge X \sharp* (object\ \alpha))$
and $(xvec \sharp* \alpha) = (xvec \sharp* (subject\ \alpha) \wedge xvec \sharp* (bn\ \alpha) \wedge xvec \sharp* (object\ \alpha))$
⟨*proof*⟩

lemma *subjectEqvt*[*eqvt*]:
fixes $p :: name\ prm$
and $\alpha :: ('a::fs\text{-}name)\ action$

shows $(p \cdot subject\ \alpha) = subject(p \cdot \alpha)$
⟨*proof*⟩

lemma *okjectEqvt*[*eqvt*]:
fixes $p :: name\ prm$
and $\alpha :: ('a::fs\text{-}name)\ action$

shows $(p \cdot object\ \alpha) = object(p \cdot \alpha)$
⟨*proof*⟩

lemma *create-residualEqvt*[*eqvt*]:
fixes $p :: name\ prm$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $(p \cdot (\alpha \prec P)) = (p \cdot \alpha) \prec (p \cdot P)$
⟨*proof*⟩

lemma *residualFresh*:
fixes $x :: name$
and $\alpha :: 'a::fs\text{-}name\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $(x \sharp (\alpha \prec P)) = (x \sharp (subject\ \alpha) \wedge (x \in (set(bn(\alpha))) \vee (x \sharp object(\alpha) \wedge x \sharp P)))$
⟨*proof*⟩

lemma *residualFresh2*[*simp*]:
fixes $x :: name$
and $\alpha :: ('a::fs\text{-}name)\ action$

and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $x \sharp \alpha$
and $x \sharp P$

shows $x \sharp \alpha \prec P$
⟨*proof*⟩

lemma *residualFreshChain2*[*simp*]:
fixes $xvec :: name\ list$
and $X :: name\ set$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $[\![xvec \sharp* \alpha; xvec \sharp* P]\!] \Longrightarrow xvec \sharp* (\alpha \prec P)$
and $[\![X \sharp* \alpha; X \sharp* P]\!] \Longrightarrow X \sharp* (\alpha \prec P)$
⟨*proof*⟩

lemma *residualFreshSimp*[*simp*]:
fixes $x :: name$
and $M :: 'a::fs\text{-}name$
and $N :: 'a$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $x \sharp (M(\!|N|\!) \prec P) = (x \sharp M \wedge x \sharp N \wedge x \sharp P)$
and $x \sharp (M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P) = (x \sharp M \wedge x \sharp ((\!|\nu{*}xvec|\!)(N \prec' P)))$
and $x \sharp (\tau \prec P) = (x \sharp P)$
⟨*proof*⟩

lemma *residualInject'*:

shows $(\alpha \prec P = RIn\ M\ N\ Q) = (P = Q \wedge \alpha = M(\!|N|\!))$
and $(\alpha \prec P = ROut\ M\ B) = (\exists\, xvec\ N.\ \alpha = M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
and $(\alpha \prec P = RTau\ Q) = (\alpha = \tau \wedge P = Q)$
and $(RIn\ M\ N\ Q = \alpha \prec P) = (P = Q \wedge \alpha = M(\!|N|\!))$
and $(ROut\ M\ B = \alpha \prec P) = (\exists\, xvec\ N.\ \alpha = M(\!|\nu{*}xvec|\!)\langle N\rangle \wedge B = (\!|\nu{*}xvec|\!)(N \prec' P))$
and $(RTau\ Q = \alpha \prec P) = (\alpha = \tau \wedge P = Q)$
⟨*proof*⟩

lemma *residualFreshChainSimp*[*simp*]:
fixes $xvec :: name\ list$
and $X :: name\ set$
and $M :: 'a::fs\text{-}name$
and $N :: 'a$
and $yvec :: name\ list$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $xvec \sharp* (M\langle\!|N|\!\rangle \prec P) = (xvec \sharp* M \wedge xvec \sharp* N \wedge xvec \sharp* P)$
and $xvec \sharp* (M\langle\!|\nu*yvec|\!\rangle\langle N\rangle \prec P) = (xvec \sharp* M \wedge xvec \sharp* (\langle\!|\nu*yvec|\!\rangle(N \prec' P)))$
and $xvec \sharp* (\tau \prec P) = (xvec \sharp* P)$
and $X \sharp* (M\langle\!|N|\!\rangle \prec P) = (X \sharp* M \wedge X \sharp* N \wedge X \sharp* P)$
and $X \sharp* (M\langle\!|\nu*yvec|\!\rangle\langle N\rangle \prec P) = (X \sharp* M \wedge X \sharp* (\langle\!|\nu*yvec|\!\rangle(N \prec' P)))$
and $X \sharp* (\tau \prec P) = (X \sharp* P)$
⟨*proof*⟩

lemma *residualFreshChainSimp2*[*simp*]:
fixes $xvec :: name\ list$
and $X :: name\ set$
and $M :: 'a::fs\text{-}name$
and $N :: 'a$
and $yvec :: name\ list$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

shows $xvec \sharp* (RIn\ M\ N\ P) = (xvec \sharp* M \wedge xvec \sharp* N \wedge xvec \sharp* P)$
and $xvec \sharp* (ROut\ M\ B) = (xvec \sharp* M \wedge xvec \sharp* B)$
and $xvec \sharp* (RTau\ P) = (xvec \sharp* P)$
and $X \sharp* (RIn\ M\ N\ P) = (X \sharp* M \wedge X \sharp* N \wedge X \sharp* P)$
and $X \sharp* (ROut\ M\ B) = (X \sharp* M \wedge X \sharp* B)$
and $X \sharp* (RTau\ P) = (X \sharp* P)$
⟨*proof*⟩

lemma *freshResidual3*[*dest*]:
fixes $x :: name$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $x \sharp bn\ \alpha$
and $x \sharp \alpha \prec P$

shows $x \sharp \alpha$ **and** $x \sharp P$
⟨*proof*⟩

lemma *freshResidualChain3*[*dest*]:
fixes $xvec :: name\ list$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $xvec \sharp* (\alpha \prec P)$
and $xvec \sharp* bn\ \alpha$

shows $xvec \sharp* \alpha$ **and** $xvec \sharp* P$
⟨*proof*⟩

lemma *freshResidual4*[*dest*]:

fixes $x :: name$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $x \sharp \alpha \prec P$

shows $x \sharp subject\ \alpha$
⟨*proof*⟩

lemma *freshResidualChain4*[*dest*]:
fixes $xvec :: name\ list$
and $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$

assumes $xvec \sharp* (\alpha \prec P)$

shows $xvec \sharp* subject\ \alpha$
⟨*proof*⟩

lemma *alphaOutputResidual*:
fixes $M :: 'a::fs\text{-}name$
and $xvec :: name\ list$
and $N :: 'a$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $p :: name\ prm$

assumes $(p \cdot xvec) \sharp* N$
and $(p \cdot xvec) \sharp* P$
and $set\ p \subseteq set\ xvec \times set(p \cdot xvec)$
and $set\ xvec \subseteq set\ yvec$

shows $M\langle\!|\nu*yvec|\!\rangle\langle N\rangle \prec P = M\langle\!|\nu*(p \cdot yvec)|\!\rangle\langle(p \cdot N)\rangle \prec (p \cdot P)$
⟨*proof*⟩

lemmas[*simp del*] = *create-residual.simps*

lemma *residualInject''*:

assumes $bn\ \alpha = bn\ \beta$

shows $(\alpha \prec P = \beta \prec Q) = (\alpha = \beta \wedge P = Q)$
⟨*proof*⟩

lemmas *residualInject* = *residual.inject create-residual.simps residualInject' residualInject''*

lemma *bnFreshResidual*[*simp*]:
fixes $\alpha :: ('a::fs\text{-}name)\ action$

shows $(bn\ \alpha) \sharp* (\alpha \prec P) = bn\ \alpha \sharp* (subject\ \alpha)$
⟨*proof*⟩

lemma *actionCases*[*case-names cInput cOutput cTau*]:
fixes $\alpha :: ('a::fs\text{-}name)\ action$

assumes $\bigwedge M\ N.\ \alpha = M(\!|N|\!) \Longrightarrow Prop$
and $\bigwedge M\ xvec\ N.\ \alpha = M(\!|\nu * xvec|\!)\langle N\rangle \Longrightarrow Prop$
and $\alpha = \tau \Longrightarrow Prop$

shows *Prop*
⟨*proof*⟩

lemma *actionPar1Dest*:
fixes $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $\beta :: ('a::fs\text{-}name)\ action$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $bn\ \alpha \sharp* bn\ \beta$

obtains $T\ p$ **where** $set\ p \subseteq set(bn\ \alpha) \times set(bn\ \beta)$ **and** $P = T \parallel (p \cdot R)$ **and** $\alpha \prec T = \beta \prec Q$
⟨*proof*⟩

lemma *actionPar2Dest*:
fixes $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $\beta :: ('a::fs\text{-}name)\ action$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $bn\ \alpha \sharp* bn\ \beta$

obtains $T\ p$ **where** $set\ p \subseteq set(bn\ \alpha) \times set(bn\ \beta)$ **and** $P = (p \cdot Q) \parallel T$ **and** $\alpha \prec T = \beta \prec R$
⟨*proof*⟩

lemma *actionScopeDest*:
fixes $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
fixes $\beta :: ('a::fs\text{-}name)\ action$
and $x :: name$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\alpha \prec P = \beta \prec (\!|\nu x|\!)\, Q$

and $x \sharp bn\ \alpha$
and $x \sharp bn\ \beta$

obtains R **where** $P = (\!|\nu x|\!)R$ **and** $\alpha \prec R = \beta \prec Q$
⟨*proof*⟩

abbreviation
outputJudge (‹-⟨-⟩› [110, 110] 110) **where** $M\langle N\rangle \equiv M(\!|\nu*([])|\!)\langle N\rangle$

declare [[*unify-trace-bound*=100]]

locale *env* = *substPsi substTerm substAssert substCond* +
assertion SCompose' SImp' SBottom' SChanEq'
for *substTerm* :: ('a::*fs-name*) ⇒ *name list* ⇒ 'a::*fs-name list* ⇒ 'a
and *substAssert* :: ('b::*fs-name*) ⇒ *name list* ⇒ 'a::*fs-name list* ⇒ 'b
and *substCond* :: ('c::*fs-name*) ⇒ *name list* ⇒ 'a::*fs-name list* ⇒ 'c
and *SCompose'* :: 'b ⇒ 'b ⇒ 'b
and *SImp'* :: 'b ⇒ 'c ⇒ *bool*
and *SBottom'* :: 'b
and *SChanEq'* :: 'a ⇒ 'a ⇒ 'c
begin
notation *SCompose'* (**infixr** ‹⊗› 90)
notation *SImp'* (‹- ⊢ -› [85, 85] 85)
notation *FrameImp* (‹- ⊢$_F$ -› [85, 85] 85)
abbreviation
FBottomJudge (‹⊥$_F$› 90) **where** $\bot_F \equiv$ (*FAssert SBottom'*)
notation *SChanEq'* (‹- ↔ -› [90, 90] 90)
notation *substTerm* (‹-[-::=-]› [100, 100, 100] 100)
notation *subs* (‹-[-::=-]› [100, 100, 100] 100)
notation *AssertionStatEq* (‹- ≃ -› [80, 80] 80)
notation *FrameStatEq* (‹- ≃$_F$ -› [80, 80] 80)
notation *SBottom'* (‹**1**› 190)
abbreviation *insertAssertion'* (‹*insertAssertion*›) **where** *insertAssertion'* ≡ *assertionAux.insertAssertion* (⊗)

inductive *semantics* :: 'b ⇒ ('a, 'b, 'c) *psi* ⇒ ('a, 'b, 'c) *residual* ⇒ *bool*
(‹- ▷ - ⟼ -› [50, 50, 50] 50)
where
cInput: ⟦Ψ ⊢ $M \leftrightarrow K$; *distinct xvec*; *set xvec* ⊆ *supp N*; *xvec* ♯* *Tvec*;
length xvec = *length Tvec*;
xvec ♯* Ψ; *xvec* ♯* M; *xvec* ♯* K⟧ ⟹ Ψ ▷ $M(\!|\lambda*xvec\ N|\!).P \longmapsto K(\!|(N[xvec::=Tvec])|\!) \prec P[xvec::=Tvec]$
| *Output*: ⟦Ψ ⊢ $M \leftrightarrow K$⟧ ⟹ Ψ ▷ $M\langle N\rangle.P \longmapsto K\langle N\rangle \prec P$
| *Case*: ⟦Ψ ▷ $P \longmapsto Rs$; (φ, P) *mem Cs*; Ψ ⊢ φ; *guarded P*⟧ ⟹ Ψ ▷ *Cases Cs* $\longmapsto Rs$

| *cPar1*: ⟦(Ψ ⊗ Ψ_Q) ▷ $P \longmapsto \alpha \prec P'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
A_Q ♯* P; A_Q ♯* Q; A_Q ♯* Ψ; A_Q ♯* α; A_Q ♯* P'; *distinct*(*bn* α);
bn α ♯* Ψ; *bn* α ♯* Ψ_Q; *bn* α ♯* Q; *bn* α ♯* P; *bn* α ♯* (*subject* α)⟧ ⟹

$\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P' \parallel Q)$

| *cPar2*: $[\![(\Psi \otimes \Psi_P) \rhd Q \longmapsto \alpha \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* \alpha$; $A_P \sharp* Q'$; *distinct*$(bn\ \alpha)$; $bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* \Psi_P$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* Q$; $bn\ \alpha \sharp* (subject\ \alpha)]\!] \Longrightarrow$ $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$

| *cComm1*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P; $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q; $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; $A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* K$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; *distinct* $xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* Q$; $xvec \sharp* K]\!] \Longrightarrow$ $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$

| *cComm2*: $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P; $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q; $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; $A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* K$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$; *distinct* $xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* Q$; $xvec \sharp* K]\!] \Longrightarrow$ $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$

| *cOpen*: $[\![\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $x \sharp xvec$; $x \sharp yvec$; $x \sharp M$; $x \sharp \Psi$; *distinct* $xvec$; *distinct* $yvec$; $xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* yvec$; $yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M]\!] \Longrightarrow$ $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu * (xvec @ x \# yvec)|\!)\langle N\rangle \prec P'$

| *cScope*: $[\![\Psi \rhd P \longmapsto \alpha \prec P'$; $x \sharp \Psi$; $x \sharp \alpha$; $bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* (subject\ \alpha)$; *distinct*$(bn\ \alpha)]\!] \Longrightarrow \Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec ((\!|\nu x|\!)P')$

| *Bang*: $[\![\Psi \rhd P \parallel\ !P \longmapsto Rs$; *guarded* $P]\!] \Longrightarrow \Psi \rhd\ !P \longmapsto Rs$

abbreviation

semanticsBottomJudge ($\langle$- $\longmapsto$ -$\rangle$ [50, 50] 50) **where** $P \longmapsto Rs \equiv \mathbf{1} \rhd P \longmapsto Rs$

equivariance *env.semantics*

nominal-inductive2 *env.semantics*

avoids *cInput*: *set xvec*

| *cPar1*: *set* $A_Q \cup$ *set*$(bn\ \alpha)$

| *cPar2*: *set* $A_P \cup$ *set*$(bn\ \alpha)$

| *cComm1*: *set* $A_P \cup$ *set* $A_Q \cup$ *set xvec*
| *cComm2*: *set* $A_P \cup$ *set* $A_Q \cup$ *set xvec*
| *cOpen*: $\{x\} \cup$ *set xvec* $\cup$ *set yvec*
| *cScope*: $\{x\} \cup set(bn\ \alpha)$
⟨*proof*⟩

lemma *nilTrans*[*dest*]:
fixes Ψ :: $'b$
and *Rs* :: $('a, 'b, 'c)$ *residual*
and *M* :: $'a$
and *xvec* :: *name list*
and *N* :: $'a$
and *P* :: $('a, 'b, 'c)$ *psi*
and *K* :: $'a$
and *yvec* :: *name list*
and N' :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and *CsP* :: $('c \times ('a, 'b, 'c)\ psi)\ list$
and Ψ' :: $'b$

shows $\Psi \rhd \mathbf{0} \longmapsto Rs \Longrightarrow False$
and $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto K(\!|\nu * yvec|\!)\langle N'\rangle \prec P' \Longrightarrow False$
and $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \tau \prec P' \Longrightarrow False$
and $\Psi \rhd M\langle N\rangle.P \longmapsto K(\!|N'|\!) \prec P' \Longrightarrow False$
and $\Psi \rhd M\langle N\rangle.P \longmapsto \tau \prec P' \Longrightarrow False$
and $\Psi \rhd \{\!|\Psi'|\!\} \longmapsto Rs \Longrightarrow False$
⟨*proof*⟩

lemma *residualEq*:
fixes α :: $'a$ *action*
and *P* :: $('a, 'b, 'c)$ *psi*
and β :: $'a$ *action*
and *Q* :: $('a, 'b, 'c)$ *psi*

assumes $\alpha \prec P = \beta \prec Q$
and $bn\ \alpha \sharp* (bn\ \beta)$
and $distinct(bn\ \alpha)$
and $distinct(bn\ \beta)$
and $bn\ \alpha \sharp* (\alpha \prec P)$
and $bn\ \beta \sharp* (\beta \prec Q)$

obtains *p* **where** *set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **and** *distinctPerm p* **and** $\beta = p \cdot \alpha$ **and** $Q = p \cdot P$ **and** $bn\ \alpha \sharp* \beta$ **and** $bn\ \alpha \sharp* Q$ **and** $bn(p \cdot \alpha) \sharp* \alpha$ **and** $bn(p \cdot \alpha) \sharp* P$
⟨*proof*⟩

lemma *semanticsInduct*[*consumes 3*, *case-names cAlpha cInput cOutput cCase cPar1 cPar2 cComm1 cComm2 cOpen cScope cBang*]:
fixes Ψ :: $'b$

and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'d$::*fs-name* $\Rightarrow$ $'b$ $\Rightarrow$ $('a, 'b, 'c)$ *psi* $\Rightarrow$
$'a$ *action* $\Rightarrow$ $('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
and C :: $'d$::*fs-name*

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and *bn* $\alpha \sharp* (subject\ \alpha)$
and *distinct*(*bn* α)
and *rAlpha*: $\bigwedge \Psi\ P\ \alpha\ P'\ p\ C.\ [\![bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* (subject\ \alpha);$
$bn\ \alpha \sharp* C;\ bn\ \alpha \sharp* (bn(p \cdot \alpha));$
$set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha));\ distinctPerm\ p;$
$(bn(p \cdot \alpha)) \sharp* \alpha;\ (bn(p \cdot \alpha)) \sharp* P';\ Prop\ C\ \Psi\ P\ \alpha\ P'$
$]\!] \Longrightarrow$
$Prop\ C\ \Psi\ P\ (p \cdot \alpha)\ (p \cdot P')$
and *rInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash M \leftrightarrow K;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;\ xvec \sharp* \Psi;$
$xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$(K(\!|(N[xvec::=Tvec])|\!))\ (P[xvec::=Tvec])$
and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ [\![\Psi \vdash M \leftrightarrow K]\!] \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)$
$(K\langle N\rangle)\ P$
and *rCase*: $\bigwedge \Psi\ P\ \alpha\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto \alpha \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P';$
$(\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ \alpha\ P'$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct$
$A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ \alpha\ P';$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* P';\ A_Q \sharp* C;$
$distinct(bn\ \alpha);\ bn\ \alpha \sharp* Q;$
$bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* \Psi_Q;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C]\!]$
$\Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P' \parallel Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct$
$A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ \alpha\ Q';$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* Q';\ A_P \sharp* C;$
$distinct(bn\ \alpha);\ bn\ \alpha \sharp* Q;$
$bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* \Psi_P;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C]\!]$
$\Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P \parallel Q')$
and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|N|\!))$
$P';$
$extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$

$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec Q'$; $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_P)$ Q $(K(\!|\nu{*}xvec|\!)\langle N\rangle)$ Q';
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$;
$A_P \sharp{*} \Psi$; $A_P \sharp{*} \Psi_Q$; $A_P \sharp{*} P$; $A_P \sharp{*} M$; $A_P \sharp{*} N$; $A_P \sharp{*} P'$;
$A_P \sharp{*} Q$; $A_P \sharp{*} Q'$; $A_P \sharp{*} A_Q$; $A_P \sharp{*} xvec$; $A_Q \sharp{*} \Psi$; $A_Q \sharp{*} \Psi_P$;
$A_Q \sharp{*} P$; $A_Q \sharp{*} N$; $A_Q \sharp{*} P'$; $A_Q \sharp{*} Q$; $A_Q \sharp{*} K$; $A_Q \sharp{*} Q'$; *distinct xvec*;
$A_Q \sharp{*} xvec$; $xvec \sharp{*} \Psi$; $xvec \sharp{*} \Psi_P$; $xvec \sharp{*} \Psi_Q$; $xvec \sharp{*} P$; $xvec \sharp{*} M$;
$xvec \sharp{*} Q$; $xvec \sharp{*} K$; $A_P \sharp{*} C$; $A_Q \sharp{*} C$; $xvec \sharp{*} C$⟧ ⟹
Prop C Ψ $(P \parallel Q)$ (τ) $((\!|\nu{*}xvec|\!)(P' \parallel Q'))$

and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C$.
⟦$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'$; $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_Q)$ P $(M(\!|\nu{*}xvec|\!)\langle N\rangle)$ P';
extractFrame $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; $\bigwedge C$. *Prop* C $(\Psi \otimes \Psi_P)$ Q $(K(\!|N|\!))$ Q';
extractFrame $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$;
$A_P \sharp{*} \Psi$; $A_P \sharp{*} \Psi_Q$; $A_P \sharp{*} P$; $A_P \sharp{*} M$; $A_P \sharp{*} N$; $A_P \sharp{*} P'$;
$A_P \sharp{*} Q$; $A_P \sharp{*} Q'$; $A_P \sharp{*} A_Q$; $A_P \sharp{*} xvec$; $A_Q \sharp{*} \Psi$; $A_Q \sharp{*} \Psi_P$;
$A_Q \sharp{*} P$; $A_Q \sharp{*} N$; $A_Q \sharp{*} P'$; $A_Q \sharp{*} Q$; $A_Q \sharp{*} K$; $A_Q \sharp{*} Q'$; *distinct xvec*;
$A_Q \sharp{*} xvec$; $xvec \sharp{*} \Psi$; $xvec \sharp{*} \Psi_P$; $xvec \sharp{*} \Psi_Q$; $xvec \sharp{*} P$; $xvec \sharp{*} M$;
$xvec \sharp{*} Q$; $xvec \sharp{*} K$; $A_P \sharp{*} C$; $A_Q \sharp{*} C$; $xvec \sharp{*} C$⟧ ⟹
Prop C Ψ $(P \parallel Q)$ (τ) $((\!|\nu{*}xvec|\!)(P' \parallel Q'))$

and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C$.
⟦$\Psi \rhd P \longmapsto M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle \prec P'$; $x \in supp\ N$; $\bigwedge C$. *Prop* C Ψ P $(M(\!|\nu{*}(xvec@yvec)|\!)\langle N\rangle)$ P';
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp yvec$; $xvec \sharp{*} \Psi$; $xvec \sharp{*} P$; $xvec \sharp{*} M$; *distinct xvec*; *distinct yvec*;
$yvec \sharp{*} \Psi$; $yvec \sharp{*} P$; $yvec \sharp{*} M$; $yvec \sharp{*} C$; $x \sharp C$; $xvec \sharp{*} C$⟧ ⟹
Prop C Ψ $((\!|\nu x|\!)P)$ $(M(\!|\nu{*}(xvec@x\#yvec)|\!)\langle N\rangle)$ P'

and *rScope*: $\bigwedge \Psi\ P\ \alpha\ P'\ x\ C$.
⟦$\Psi \rhd P \longmapsto \alpha \prec P'$; $\bigwedge C$. *Prop* C Ψ P α P';
$x \sharp \Psi$; $x \sharp \alpha$; $bn\ \alpha \sharp{*} \Psi$;
$bn\ \alpha \sharp{*} P$; $bn\ \alpha \sharp{*} (subject\ \alpha)$; $x \sharp C$; $bn\ \alpha \sharp{*} C$; *distinct*$(bn\ \alpha)$⟧ ⟹
Prop C Ψ $((\!|\nu x|\!)P)$ α $((\!|\nu x|\!)P')$

and *rBang*: $\bigwedge \Psi\ P\ \alpha\ P'\ C$.
⟦$\Psi \rhd P \parallel !P \longmapsto \alpha \prec P'$; *guarded* P; $\bigwedge C$. *Prop* C Ψ $(P \parallel !P)$ α P'⟧ ⟹
Prop C Ψ $(!P)$ α P'

shows *Prop* C Ψ P α P'
⟨*proof*⟩

lemma *outputInduct*[*consumes 1*, *case-names cOutput cCase cPar1 cPar2 cOpen cScope cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and *Prop* :: $'d$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ $'a \Rightarrow ('a, 'b, 'c)$ *boundOutput* $\Rightarrow$ *bool*
and C :: $'d$::*fs-name*

assumes $\Psi \rhd P \longmapsto ROut\ M\ B$
and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ [\![\Psi \vdash M \leftrightarrow K]\!] \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)\ K\ (N \prec' P)$
and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ C.$
$[\![\Psi \rhd P \longmapsto (ROut\ M\ B); \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B; (\varphi, P)\ mem\ Cs; \Psi \vdash \varphi; guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ \ P'\ A_Q\ Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'; extractFrame\ Q = \langle A_Q, \Psi_Q\rangle; distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ ((\!|\nu * xvec|\!)N \prec' P');$
$A_Q \sharp* P; A_Q \sharp* Q; A_Q \sharp* \Psi; A_Q \sharp* M;$
$A_Q \sharp* xvec; A_Q \sharp* N; A_Q \sharp* P'; A_Q \sharp* C; xvec \sharp* Q;$
$xvec \sharp* \Psi; xvec \sharp* \Psi_Q; xvec \sharp* P; xvec \sharp* M; xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P' \parallel Q))$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ xvec\ N\ \ Q'\ A_P\ P\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec Q'; extractFrame\ P = \langle A_P, \Psi_P\rangle; distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\!|\nu * xvec|\!)N \prec' Q');$
$A_P \sharp* P; A_P \sharp* Q; A_P \sharp* \Psi; A_P \sharp* M;$
$A_P \sharp* xvec; A_P \sharp* N; A_P \sharp* Q'; A_P \sharp* C; xvec \sharp* P;$
$xvec \sharp* \Psi; xvec \sharp* \Psi_P; xvec \sharp* Q; xvec \sharp* M; xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P \parallel Q'))$
and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P'; x \in supp\ N; \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ ((\!|\nu * (xvec @ yvec)|\!)N \prec' P');$
$x \sharp \Psi; x \sharp M; x \sharp xvec; x \sharp yvec; xvec \sharp* \Psi; xvec \sharp* P; xvec \sharp* M;$
$xvec \sharp* yvec; yvec \sharp* \Psi; yvec \sharp* P; yvec \sharp* M; yvec \sharp* C; x \sharp C; xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ ((\!|\nu * (xvec @ x \# yvec)|\!)N \prec' P')$
and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'; \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ ((\!|\nu * xvec|\!)N \prec' P');$
$x \sharp \Psi; x \sharp M; x \sharp xvec; x \sharp N; xvec \sharp* \Psi; xvec \sharp* P; xvec \sharp* M;$
$x \sharp C; xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ ((\!|\nu * xvec|\!)N \prec' (\!|\nu x|\!)P')$
and *rBang*: $\bigwedge \Psi\ P\ M\ B\ C.$
$[\![\Psi \rhd P \parallel !P \longmapsto (ROut\ M\ B); guarded\ P; \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ M\ B]\!] \Longrightarrow$

$Prop\ C\ \Psi\ (!P)\ M\ B$
shows $Prop\ C\ \Psi\ P\ M\ B$
⟨*proof*⟩

lemma *boundOutputBindObject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and M :: $'a$
and $yvec$ :: $name\ list$
and N :: $'a$
and P' :: $('a, 'b, 'c)\ psi$
and y :: $name$

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and $bn\ \alpha \sharp* subject\ \alpha$
and $distinct(bn\ \alpha)$
and $y \in set(bn\ \alpha)$

shows $y \in supp(object\ \alpha)$
⟨*proof*⟩

lemma *alphaBoundOutputChain'*:
fixes $yvec$ :: $name\ list$
and $xvec$ :: $name\ list$
and B :: $('a, 'b, 'c)\ boundOutput$

assumes $length\ xvec = length\ yvec$
and $yvec \sharp* B$
and $yvec \sharp* xvec$
and $distinct\ yvec$

shows $(\!|\nu * xvec|\!)B = (\!|\nu * yvec|\!)([xvec\ yvec] \cdot_v B)$
⟨*proof*⟩

lemma *alphaBoundOutputChain''*:
fixes $yvec$ :: $name\ list$
and $xvec$ :: $name\ list$
and N :: $'a$
and P :: $('a, 'b, 'c)\ psi$

assumes $length\ xvec = length\ yvec$
and $yvec \sharp* N$
and $yvec \sharp* P$
and $yvec \sharp* xvec$
and $distinct\ yvec$

shows $(\!|\nu * xvec|\!)(N \prec' P) = (\!|\nu * yvec|\!)(([xvec\ yvec] \cdot_v N) \prec' ([xvec\ yvec] \cdot_v P))$
⟨*proof*⟩

lemma *alphaDistinct*:
fixes *xvec* :: *name list*
and *N* :: *'a*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *yvec* :: *name list*
and *M* :: *'a*
and *Q* :: (*'a*, *'b*, *'c*) *psi*

assumes $\alpha \prec P = \beta \prec Q$
and *distinct*(*bn* α)
and $\bigwedge x.\ x \in set(bn\ \alpha) \Longrightarrow x \in supp(object\ \alpha)$
and *bn* α $\sharp*$ *bn* β
and *bn* α $\sharp*$ (*object* β)
and *bn* α $\sharp*$ *Q*

shows *distinct*(*bn* β)
⟨*proof*⟩

lemma *boundOutputDistinct*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and α :: *'a action*
and *P'* :: (*'a*, *'b*, *'c*) *psi*

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

shows *distinct*(*bn* α)
⟨*proof*⟩

lemma *inputDistinct*:
fixes Ψ :: *'b*
and *M* :: *'a*
and *xvec* :: *name list*
and *N* :: *'a*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Rs* :: (*'a*, *'b*, *'c*) *residual*

assumes $\Psi \rhd M(\!|\lambda* xvec\ N|\!).P \longmapsto Rs$

shows *distinct xvec*
⟨*proof*⟩

lemma *outputInduct'*[*consumes 2*, *case-names cAlpha cOutput cCase cPar1 cPar2 cOpen cScope cBang*]:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *M* :: *'a*
and *yvec* :: *name list*
and *N* :: *'a*

and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'d$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$
$'a \Rightarrow$ *name list* $\Rightarrow$ $'a \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
and C :: $'d$::*fs-name*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and $xvec \sharp* M$
and *rAlpha*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ p\ C.\ [\![xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C;\ xvec \sharp* (p \cdot xvec);$
$set\ p \subseteq set\ xvec \times set(p \cdot xvec);\ distinctPerm\ p;$
$(p \cdot xvec) \sharp* N;\ (p \cdot xvec) \sharp* P';\ Prop\ C\ \Psi\ P\ M\ xvec\ N\ P']\!] \Longrightarrow$
$Prop\ C\ \Psi\ P\ M\ (p \cdot xvec)\ (p \cdot N)\ (p \cdot P')$
and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ [\![\Psi \vdash M \leftrightarrow K]\!] \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)\ K\ ([])\ N\ P$
and *rCase*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ M\ xvec\ N\ P';\ (\varphi, P) \in mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ xvec\ N\ P'$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ xvec\ N\ P';$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* M;$
$A_Q \sharp* xvec;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* C;\ xvec \sharp* Q;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P' \parallel Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ xvec\ N\ Q'\ A_P\ P\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ xvec\ N\ Q';$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* M;$
$A_P \sharp* xvec;\ A_P \sharp* N;\ A_P \sharp* Q';\ A_P \sharp* C;\ xvec \sharp* Q;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ xvec\ N\ (P \parallel Q')$
and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P';\ x \in supp\ N;\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ (xvec @ yvec)\ N\ P';$
$x \sharp \Psi;\ x \sharp M;\ x \sharp xvec;\ x \sharp yvec;\ xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;$
$yvec \sharp* \Psi;\ yvec \sharp* P;\ yvec \sharp* M;\ yvec \sharp* C;\ x \sharp C;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ (xvec @ x \# yvec)\ N\ P'$
and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ xvec\ N\ P';$
$x \sharp \Psi;\ x \sharp M;\ x \sharp xvec;\ x \sharp N;\ xvec \sharp* \Psi;$
$xvec \sharp* P;\ xvec \sharp* M;\ x \sharp C;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ xvec\ N\ ((\!|\nu x|\!)P')$
and *rBang*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ C.$
$[\![\Psi \rhd P \parallel\ !P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ guarded\ P;\ \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel\ !P)\ M\ xvec\ N\ P']\!] \Longrightarrow$
$Prop\ C\ \Psi\ (!P)\ M\ xvec\ N\ P'$

shows *Prop C* Ψ *P M xvec N P'*
⟨*proof*⟩

lemma *inputInduct*[*consumes 1*, *case-names cInput cCase cPar1 cPar2 cScope cBang*]:
fixes Ψ :: *'b*
and *P* :: *('a, 'b, 'c) psi*
and *M* :: *'a*
and *N* :: *'a*
and *P'* :: *('a, 'b, 'c) psi*
and *Prop* :: *'d::fs-name* ⇒ *'b* ⇒ *('a, 'b, 'c) psi* ⇒
'a ⇒ *'a* ⇒ *('a, 'b, 'c) psi* ⇒ *bool*
and *C* :: *'d::fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *rInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash M \leftrightarrow K;\ distinct\ xvec;\ set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec;\ xvec \sharp* \Psi;$
$xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$K\ (N[xvec ::= Tvec])\ (P[xvec ::= Tvec])$
and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P';\ (\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ M\ N\ P'$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_Q\ Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ N\ P';\ distinct\ A_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* M;\ A_Q \sharp* N;$
$A_Q \sharp* P';\ A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P' \parallel Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ N\ Q'\ A_P\ P\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ N\ Q';\ distinct\ A_P;$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* M;\ A_P \sharp* N;$
$A_P \sharp* Q';\ A_P \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P \parallel Q')$
and *rScope*: $\bigwedge \Psi\ P\ M\ N\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P';\ x \sharp \Psi;\ x \sharp M;\ x \sharp N;\ x \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ N\ ((\!|\nu x|\!)P')$
and *rBang*: $\bigwedge \Psi\ P\ M\ N\ P'\ C.$
$[\![\Psi \rhd P \parallel\ !P \longmapsto M(\!|N|\!) \prec P';\ guarded\ P;\ \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel\ !P)\ M\ N\ P']\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ M\ N\ P'$
shows *Prop C* Ψ *P M N P'*
⟨*proof*⟩

lemma *tauInduct*[*consumes 1*, *case-names cCase cPar1 cPar2 cComm1 cComm2*

cScope cBang]:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Rs :: (*'a*, *'b*, *'c*) *residual*
and *Prop* :: *'d*::*fs-name* ⇒ *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒
(*'a*, *'b*, *'c*) *psi* ⇒ *bool*
and C :: *'d*::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto \tau \prec P'$
and *rCase*: $\bigwedge \Psi\ P\ P'\ \varphi\ Cs\ C.\ [\![\Psi \rhd P \longmapsto \tau \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ P\ P';\ (\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ P'$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ P'\ A_Q\ Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \tau \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ P';$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;$
$A_Q \sharp* P';\ A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (P' \parallel Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ Q'\ A_P\ P\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \tau \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ Q';$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;$
$A_P \sharp* Q';\ A_P \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (P \parallel Q')$
and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;$
$distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;$
$xvec \sharp* Q;\ xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ ((\!|\nu* xvec|\!)(P' \parallel Q'))$
and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P';\ \ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp*$

M;
$xvec \sharp* Q$; $xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$
and *rScope*: $\bigwedge \Psi\ P\ P'\ x\ C.$
$[\![\Psi \rhd P \longmapsto \tau \prec P'; \bigwedge C.\ Prop\ C\ \Psi\ P\ P'; x \sharp \Psi; x \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ ((\!|\nu x|\!)P')$
and *rBang*: $\bigwedge \Psi\ P\ P'\ C.$
$[\![\Psi \rhd P \parallel !P \longmapsto \tau \prec P'; guarded\ P; \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ P']\!]$
$\Longrightarrow Prop\ C\ \Psi\ (!P)\ P'$
shows $Prop\ C\ \Psi\ P\ P'$
⟨*proof*⟩

lemma *semanticsFrameInduct*[*consumes 3*, *case-names cAlpha cInput cOutput cCase cPar1 cPar2 cComm1 cComm2 cOpen cScope cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rs :: $('a, 'b, 'c)\ residual$
and A_P :: *name list*
and Ψ_P :: $'b$
and *Prop* :: $'d$::*fs-name* ⇒ $'b$ ⇒ $('a, 'b, 'c)\ psi$ ⇒
$('a, 'b, 'c)\ residual$ ⇒ *name list* ⇒ $'b$ ⇒ *bool*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto Rs$
and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and *rAlpha*: $\bigwedge \Psi\ P\ A_P\ \Psi_P\ p\ Rs\ C.\ [\![A_P \sharp* \Psi; A_P \sharp* P; A_P \sharp* (p \cdot A_P); A_P \sharp* Rs; A_P \sharp* C;$
$set\ p \subseteq set\ A_P \times set(p \cdot A_P); distinctPerm\ p;$
$Prop\ C\ \Psi\ P\ Rs\ A_P\ \Psi_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ Rs\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash M \leftrightarrow K; distinct\ xvec; set\ xvec \subseteq supp\ N;$
$length\ xvec = length\ Tvec; xvec \sharp* \Psi;$
$xvec \sharp* M; xvec \sharp* K; xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda * xvec\ N|\!).P)$
$(K(\!|(N[xvec::=Tvec])|\!) \prec (P[xvec::=Tvec]))\ ([])\ (\mathbf{1})$
and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ C\ \Psi\ (M\langle N \rangle.P)\ (K\langle N \rangle \prec P)\ ([])\ (\mathbf{1})$
and *rCase*: $\bigwedge \Psi\ P\ Rs\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto Rs; extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P; \bigwedge C.\ Prop\ C\ \Psi\ P\ Rs\ A_P\ \Psi_P;$
$(\varphi, P)\ mem\ Cs; \Psi \vdash \varphi; guarded\ P; \Psi_P \simeq \mathbf{1};$
$(supp\ \Psi_P) = (\{\}::name\ set);$
$A_P \sharp* \Psi; A_P \sharp* P; A_P \sharp* Rs; A_P \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (Cases\ Cs)\ Rs\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';$
$extractFrame\ P = \langle A_P, \Psi_P \rangle; distinct\ A_P;$
$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle; distinct\ A_Q;$

$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (\alpha \prec P')\ A_P\ \Psi_P;\ distinct(bn\ \alpha);$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* P';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* P';\ A_Q \sharp* \Psi_P;$
$bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* \Psi_P;\ bn\ \alpha \sharp* \Psi_Q;$
$A_P \sharp* C;\ A_Q \sharp* C;\ bn\ \alpha \sharp* C[\![\Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P' \parallel Q))\ (A_P@A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';$
$extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (\alpha \prec Q')\ A_Q\ \Psi_Q;\ distinct(bn\ \alpha);$
$A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* \Psi;\ A_P \sharp* \alpha;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* \Psi_Q;$
$A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* \Psi;\ A_Q \sharp* \alpha;\ A_Q \sharp* Q';\ A_Q \sharp* \Psi_P;$
$bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* \Psi_P;\ bn\ \alpha \sharp* \Psi_Q;$
$A_P \sharp* C;\ A_Q \sharp* C;\ bn\ \alpha \sharp* C[\![\Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (\alpha \prec (P \parallel Q'))\ (A_P@A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;$
$distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ ((M(\!|N|\!)) \prec P')\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|\nu * xvec|\!)\langle N \rangle \prec Q')\ A_Q\ \Psi_Q;\ distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;$
$xvec \sharp* Q;\ xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C[\![\Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P@A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|\nu * xvec|\!)\langle N \rangle \prec P')\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|N|\!) \prec Q')\ A_Q\ \Psi_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;$

$xvec \sharp* Q$; $xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$⟧ ⟹
Prop $C\ \Psi\ (P \parallel Q)\ (\tau \prec (\!|\nu* xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C$.
⟦$\Psi \rhd P \longmapsto M(\!|\nu*(xvec @ yvec)|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* $C\ \Psi\ P\ (M(\!|\nu*(xvec @ yvec)|\!)\langle N\rangle \prec P')\ A_P\ \Psi_P$; $x \in supp\ N$; $x \sharp \Psi$; $x \sharp M$;
$x \sharp A_P$; $x \sharp xvec$; $x \sharp yvec$; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* xvec$; $A_P \sharp* yvec$; $xvec \sharp* yvec$; *distinct* $xvec$; *distinct* $yvec$;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* \Psi_P$; $yvec \sharp* \Psi_P$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $A_P \sharp* C$; $x \sharp C$; $xvec \sharp* C$; $yvec \sharp* C$⟧ ⟹
Prop $C\ \Psi\ ((\!|\nu x|\!)P)\ (M(\!|\nu*(xvec @ x\#yvec)|\!)\langle N\rangle \prec P')\ (x\#A_P)\ \Psi_P$

and *rScope*: $\bigwedge \Psi\ P\ \alpha\ P'\ x\ A_P\ \Psi_P\ C$.
⟦$\Psi \rhd P \longmapsto \alpha \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* $C\ \Psi\ P\ (\alpha \prec P')\ A_P\ \Psi_P$;
$x \sharp \Psi$; $x \sharp \alpha$; $x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$;
$A_P \sharp* \alpha$; $A_P \sharp* P'$; *distinct*$(bn\ \alpha)$;
$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* \Psi_P$;
$A_P \sharp* C$; $x \sharp C$; $bn\ \alpha \sharp* C$⟧ ⟹
Prop $C\ \Psi\ ((\!|\nu x|\!)P)\ (\alpha \prec ((\!|\nu x|\!)P'))\ (x\#A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ Rs\ A_P\ \Psi_P\ C$.
⟦$\Psi \rhd P \parallel\ !P \longmapsto Rs$; *guarded* P; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* $C\ \Psi\ (P \parallel\ !P)\ Rs\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; $supp\ \Psi_P = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Rs$; $A_P \sharp* C$⟧ ⟹ *Prop* $C\ \Psi\ (!P)\ Rs\ ([])\ (\mathbf{1})$

shows *Prop* $C\ \Psi\ P\ Rs\ A_P\ \Psi_P$
⟨*proof*⟩

lemma *semanticsFrameInduct'*[*consumes 5*, *case-names cAlpha cFrameAlpha cInput cOutput cCase cPar1 cPar2 cComm1 cComm2 cOpen cScope cBang*]:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and Rs :: *('a, 'b, 'c) residual*
and A_P :: *name list*
and Ψ_P :: *'b*
and *Prop* :: *'d::fs-name* ⇒ *'b* ⇒ *('a, 'b, 'c) psi* ⇒ *'a action* ⇒
('a, 'b, 'c) psi ⇒ *name list* ⇒ *'b* ⇒ *bool*
and C :: *'d::fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto \alpha \prec P'$
and *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $bn\ \alpha \sharp* subject\ \alpha$
and *distinct*$(bn\ \alpha)$
and *rAlpha*: $\bigwedge \Psi\ P\ \alpha\ P'\ p\ A_P\ \Psi_P\ C$. ⟦$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* subject$

α; $bn\ \alpha \sharp* \Psi_P$;
$bn\ \alpha \sharp* C$; $bn\ \alpha \sharp* (p \cdot \alpha)$; $A_P \sharp* \Psi$; $A_P \sharp* P$;
$A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* C$;
$set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$; $distinctPerm\ p$;
$bn(p \cdot \alpha) \sharp* \alpha$; $(bn(p \cdot \alpha)) \sharp* P'$; $Prop\ C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P]\!] \Longrightarrow$
$Prop\ C\ \Psi\ P\ (p \cdot \alpha)\ (p \cdot P')\ A_P\ \Psi_P$

and *rFrameAlpha*: $\bigwedge \Psi\ P\ A_P\ \Psi_P\ p\ \alpha\ P'\ C.\ [\![A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* (p \cdot A_P)$; $A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* C$;
$set\ p \subseteq set\ A_P \times set(p \cdot A_P)$; $distinctPerm\ p$; $A_P \sharp* subject\ \alpha$;
$Prop\ C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ \alpha\ P'\ (p \cdot A_P)\ (p \cdot \Psi_P)$

and *rInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C.$
$[\![\Psi \vdash M \leftrightarrow K$; $distinct\ xvec$; $set\ xvec \subseteq supp\ N$;
$length\ xvec = length\ Tvec$; $xvec \sharp* \Psi$;
$xvec \sharp* M$; $xvec \sharp* K$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (M(\!|\lambda* xvec\ N|\!).P)$
$(K(\!|(N[xvec::=Tvec])|\!))\ (P[xvec::=Tvec])\ ([])\ (\mathbf{1})$

and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ C\ \Psi\ (M\langle N\rangle.P)\ (K\langle N\rangle)\ P\ ([])\ (\mathbf{1})$

and *rCase*: $\bigwedge \Psi\ P\ \alpha\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto \alpha \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$; $\bigwedge C.\ Prop\ C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$;
$(\varphi, P)\ mem\ Cs$; $\Psi \vdash \varphi$; $guarded\ P$; $\Psi_P \simeq \mathbf{1}$;
$(supp\ \Psi_P) = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ \alpha\ P'\ ([])\ (\mathbf{1})$

and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ \alpha\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$;
$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ \alpha\ P'\ A_P\ \Psi_P$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* \alpha$; $A_Q \sharp* P'$; $A_Q \sharp* \Psi_P$;
$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* Q$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* \Psi_P$; $bn\ \alpha \sharp* \Psi_Q$;
$A_P \sharp* C$; $A_Q \sharp* C$; $bn\ \alpha \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ \alpha\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$;
$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ \alpha\ Q'\ A_Q\ \Psi_Q$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* \alpha$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* \alpha$; $A_Q \sharp* Q'$; $A_Q \sharp* \Psi_P$;
$bn\ \alpha \sharp* \Psi$; $bn\ \alpha \sharp* P$; $bn\ \alpha \sharp* Q$; $bn\ \alpha \sharp* subject\ \alpha$; $bn\ \alpha \sharp* \Psi_P$;

$bn\ \alpha \sharp* \Psi_Q;$
$A_P \sharp* C;\ A_Q \sharp* C;\ bn\ \alpha \sharp* C[\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ \alpha\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and $rComm1: \bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P,\ \Psi_P\rangle;$
$distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|N|\!))\ P'\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q,$
$\Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|\nu * xvec|\!)\langle N\rangle)\ Q'\ A_Q\ \Psi_Q;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp*$
$M;$
$xvec \sharp* Q;\ xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C[\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (\tau)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and $rComm2: \bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,$
$\Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ (M(\!|\nu * xvec|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle;$
$distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ (K(\!|N|\!))\ Q'\ A_Q\ \Psi_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';$
$A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* K;\ A_Q \sharp* Q';$
$A_Q \sharp* xvec;\ xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp*$
$M;$
$xvec \sharp* Q;\ xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C[\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (\tau)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and $rOpen: \bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ y\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P,$
$\Psi_P\rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ (M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle)\ P'\ A_P\ \Psi_P;\ x \in supp$
$N;\ x \sharp \Psi;\ x \sharp M;$
$x \sharp A_P;\ x \sharp xvec;\ x \sharp yvec;\ A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp*$
$N;\ A_P \sharp* P';$
$A_P \sharp* xvec;\ A_P \sharp* yvec;\ xvec \sharp* yvec;\ distinct\ xvec;\ distinct\ yvec;$
$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* \Psi_P;$
$yvec \sharp* \Psi;\ yvec \sharp* P;\ yvec \sharp* M;\ A_P \sharp* C;\ x \sharp C;\ xvec \sharp* C;\ yvec$
$\sharp* C;$
$y \neq x;\ y \sharp \Psi;\ y \sharp P;\ y \sharp M;\ y \sharp xvec;\ y \sharp yvec;\ y \sharp N;\ y \sharp P';\ y \sharp$
$A_P;\ y \sharp \Psi_P;\ y \sharp C[\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ (M(\!|\nu * (xvec @ y \# yvec)|\!)\langle([(x,\ y)] \cdot N)\rangle)\ ([(x,$
$y)] \cdot P')\ (x \# A_P)\ \Psi_P$

and $rScope: \bigwedge \Psi\ P\ \alpha\ P'\ x\ A_P\ \Psi_P\ C.$

$[\![\Psi \rhd P \longmapsto\!\!\!\rightarrow \alpha \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$;
$x \sharp \Psi$; $x \sharp \alpha$; $x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$;
$A_P \sharp* \alpha$; $A_P \sharp* P'$;
bn $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp*$ *subject* α; *bn* $\alpha \sharp* \Psi_P$;
$A_P \sharp* C$; $x \sharp C$; *bn* $\alpha \sharp* C]\!] \Longrightarrow$
Prop $C\ \Psi\ (\langle\!(\nu x)\!\rangle P)\ \alpha\ (\langle\!(\nu x)\!\rangle P')\ (x \# A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P\ C$.
$[\![\Psi \rhd P \parallel\ !P \longmapsto\!\!\!\rightarrow \alpha \prec P'$; *guarded* P; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C$. *Prop* $C\ \Psi\ (P \parallel\ !P)\ \alpha\ P'\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; *supp* $\Psi_P = (\{\}$::*name set*$)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* \alpha$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (!P)\ \alpha\ P'\ ([])\ (\mathbf{1})$

shows *Prop* $C\ \Psi\ P\ \alpha\ P'\ A_P\ \Psi_P$

⟨*proof*⟩

lemma *inputFrameInduct*[*consumes 3*, *case-names cAlpha cInput cCase cPar1 cPar2 cScope cBang*]:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and M :: *'a*
and N :: *'a*
and P' :: *('a, 'b, 'c) psi*
and *Prop* :: *'d*::*fs-name* ⇒ *'b* ⇒ *('a, 'b, 'c) psi* ⇒
'a ⇒ *'a* ⇒ *('a, 'b, 'c) psi* ⇒ *name list* ⇒ *'b* ⇒ *bool*
and C :: *'d*::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto\!\!\!\rightarrow M\langle\!(N)\!\rangle \prec P'$
and *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and *rAlpha*: $\bigwedge \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P\ p\ C$. $[\![A_P \sharp* \Psi; A_P \sharp* P; A_P \sharp* M; A_P \sharp* N; A_P \sharp* P'; A_P \sharp* (p \cdot A_P); A_P \sharp* C$;
set $p \subseteq$ *set* $A_P \times$ *set*$(p \cdot A_P)$; *distinctPerm* p;
Prop $C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P]\!] \Longrightarrow$ *Prop* $C\ \Psi\ P\ M\ N\ P'\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rInput*: $\bigwedge \Psi\ M\ K\ xvec\ N\ Tvec\ P\ C$.
$[\![\Psi \vdash M \leftrightarrow K$; *distinct xvec*; *set xvec* $\subseteq$ *supp* N;
length xvec = *length Tvec*; *xvec* $\sharp* \Psi$;
xvec $\sharp* M$; *xvec* $\sharp* K$; *xvec* $\sharp* C]\!] \Longrightarrow$
Prop $C\ \Psi\ (M\langle\!(\lambda* xvec\ N)\!\rangle.P)$
$K\ (N[xvec ::= Tvec])\ (P[xvec ::= Tvec])\ ([])\ (\mathbf{1})$
and *rCase*: $\bigwedge \Psi\ P\ M\ N\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C$. $[\![\Psi \rhd P \longmapsto\!\!\!\rightarrow M\langle\!(N)\!\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P; $\bigwedge C$. *Prop* $C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P$;
(φ, P) *mem Cs*; $\Psi \vdash \varphi$; *guarded* P; $\Psi_P \simeq \mathbf{1}$; $(supp\ \Psi_P) = (\{\}$::*name set*$)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (Cases\ Cs)\ M\ N\ P'\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C$.

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$;
$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ N\ P'\ A_P\ \Psi_P$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* M$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* \Psi_P$;
$A_P \sharp* C$; $A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ N\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$;
$extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ N\ Q'\ A_Q\ \Psi_Q$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* M$; $A_Q \sharp* N$; $A_Q \sharp* Q'$; $A_Q \sharp* \Psi_P$;
$A_P \sharp* C$; $A_Q \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ N\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rScope*: $\bigwedge \Psi\ P\ M\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P$; $x \sharp \Psi$; $x \sharp M$; $x \sharp N$;
$x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* C$; $x \sharp C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ N\ ((\!|\nu x|\!)P')\ (x \# A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \parallel !P \longmapsto M(\!|N|\!) \prec P'$; $guarded\ P$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ M\ N\ P'\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; $(supp\ \Psi_P) = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (!P)\ M\ N\ P'\ ([])\ (\mathbf{1})$

shows $Prop\ C\ \Psi\ P\ M\ N\ P'\ A_P\ \Psi_P$

⟨*proof*⟩

lemma *outputFrameInduct*[*consumes 3*, *case-names cAlpha cOutput cCase cPar1 cPar2 cOpen cScope cBang*]:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and M :: *'a*
and B :: *('a, 'b, 'c) boundOutput*
and A_P :: *name list*
and Ψ_P :: *'b*
and *Prop* :: *'d::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒ 'a ⇒ ('a, 'b, 'c) boundOutput ⇒ name list ⇒ 'b ⇒ bool*
and C :: *'d::fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto ROut\ M\ B$
and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and $distinct\ A_P$
and *rAlpha*: $\bigwedge \Psi\ P\ M\ A_P\ \Psi_P\ p\ B\ C.\ [\![A_P \sharp^* \Psi;\ A_P \sharp^* P;\ A_P \sharp^* M;\ A_P \sharp^* (p \cdot A_P);\ A_P \sharp^* B;\ A_P \sharp^* C;$
$set\ p \subseteq set\ A_P \times set(p \cdot A_P);\ distinctPerm\ p;$
$Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ M\ B\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rOutput*: $\bigwedge \Psi\ M\ K\ N\ P\ C.\ \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ C\ \Psi\ (M\langle N \rangle.P)\ K\ (N \prec' P)\ ([])\ (\mathbf{1})$
and *rCase*: $\bigwedge \Psi\ P\ M\ B\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto (ROut\ M\ B);\ extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;\ \bigwedge C.\ Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P;$
$(\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P;\ \Psi_P \simeq \mathbf{1};\ (supp\ \Psi_P) = (\{\}::name\ set);$
$A_P \sharp^* \Psi;\ A_P \sharp^* P;\ A_P \sharp^* M;\ A_P \sharp^* B;\ A_P \sharp^* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ M\ B\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec P';$
$extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ M\ ((\!|\nu * xvec|\!)N \prec' P')\ A_P\ \Psi_P;$
$A_P \sharp^* P;\ A_P \sharp^* Q;\ A_P \sharp^* \Psi;\ A_P \sharp^* M;\ A_P \sharp^* xvec;\ A_P \sharp^* N;\ A_P \sharp^* P';\ A_P \sharp^* A_Q;\ A_P \sharp^* \Psi_Q;$
$A_Q \sharp^* P;\ A_Q \sharp^* Q;\ A_Q \sharp^* \Psi;\ A_Q \sharp^* M;\ A_Q \sharp^* xvec;\ A_Q \sharp^* N;\ A_Q \sharp^* P';\ A_Q \sharp^* \Psi_P;$
$xvec \sharp^* \Psi;\ xvec \sharp^* P;\ xvec \sharp^* Q;\ xvec \sharp^* M;\ xvec \sharp^* \Psi_P;\ xvec \sharp^* \Psi_Q;$
$A_P \sharp^* C;\ A_Q \sharp^* C;\ xvec \sharp^* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P' \parallel Q))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ M\ xvec\ N\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$[\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec Q';$
$extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ M\ ((\!|\nu * xvec|\!)N \prec' Q')\ A_Q\ \Psi_Q;$
$A_P \sharp^* P;\ A_P \sharp^* Q;\ A_P \sharp^* \Psi;\ A_P \sharp^* M;\ A_P \sharp^* xvec;\ A_P \sharp^* N;\ A_P \sharp^* Q';\ A_P \sharp^* A_Q;\ A_P \sharp^* \Psi_Q;$
$A_Q \sharp^* P;\ A_Q \sharp^* Q;\ A_Q \sharp^* \Psi;\ A_Q \sharp^* M;\ A_Q \sharp^* xvec;\ A_Q \sharp^* N;\ A_Q \sharp^* Q';\ A_Q \sharp^* \Psi_P;$
$xvec \sharp^* \Psi;\ xvec \sharp^* P;\ xvec \sharp^* Q;\ xvec \sharp^* M;\ xvec \sharp^* \Psi_P;\ xvec \sharp^* \Psi_Q;$
$A_P \sharp^* C;\ A_Q \sharp^* C;\ xvec \sharp^* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ M\ ((\!|\nu * xvec|\!)N \prec' (P \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$
and *rOpen*: $\bigwedge \Psi\ P\ M\ xvec\ yvec\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N \rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$\bigwedge C.\ Prop\ C\ \Psi\ P\ M\ ((\!|\nu * (xvec @ yvec)|\!)N \prec' P')\ A_P\ \Psi_P;\ x \in supp\ N;\ x \sharp \Psi;\ x \sharp M;$
$x \sharp A_P;\ x \sharp xvec;\ x \sharp yvec;\ A_P \sharp^* \Psi;\ A_P \sharp^* P;\ A_P \sharp^* M;\ A_P \sharp^* N;\ A_P \sharp^* P';$

$A_P \sharp* xvec$; $A_P \sharp* yvec$;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* \Psi_P$;
$yvec \sharp* \Psi$; $yvec \sharp* P$; $yvec \sharp* M$; $A_P \sharp* C$; $x \sharp C$; $xvec \sharp* C$; $yvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ ((\!|\nu*(xvec@x\#yvec)|\!)N \prec' P')\ (x\#A_P)\ \Psi_P$

and *rScope*: $\bigwedge \Psi\ P\ M\ xvec\ N\ P'\ x\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ M\ ((\!|\nu*xvec|\!)N \prec' P')\ A_P\ \Psi_P$;
$x \sharp \Psi$; $x \sharp M$; $x \sharp xvec$; $x \sharp N$; $x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$;
$A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* \Psi_P$;
$A_P \sharp* C$; $x \sharp C$; $xvec \sharp* C]\!] \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ M\ ((\!|\nu*xvec|\!)N \prec' ((\!|\nu x|\!)P'))\ (x\#A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ M\ B\ A_P\ \Psi_P\ C.$
$[\![\Psi \rhd P \parallel\ !P \longmapsto ROut\ M\ B$; $guarded\ P$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel\ !P)\ M\ B\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; $supp\ \Psi_P = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ M\ B\ ([])\ (\mathbf{1})$

shows $Prop\ C\ \Psi\ P\ M\ B\ A_P\ \Psi_P$
⟨*proof*⟩

lemma *tauFrameInduct*[*consumes 3*, *case-names cAlpha cCase cPar1 cPar2 cComm1 cComm2 cScope cBang*]:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and P' :: *('a, 'b, 'c) psi*
and *Prop* :: *'d::fs-name ⇒ 'b ⇒ ('a, 'b, 'c) psi ⇒ ('a, 'b, 'c) psi ⇒ name list ⇒ 'b ⇒ bool*
and C :: *'d::fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto \tau \prec P'$
and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P\rangle$
and $distinct\ A_P$
and *rAlpha*: $\bigwedge \Psi\ P\ P'\ A_P\ \Psi_P\ p\ C.\ [\![A_P \sharp* \Psi; A_P \sharp* P; A_P \sharp* P'; A_P \sharp* (p \cdot A_P); A_P \sharp* C;$
$set\ p \subseteq set\ A_P \times set\ (p \cdot A_P)$; $distinctPerm\ p$;
$Prop\ C\ \Psi\ P\ P'\ A_P\ \Psi_P]\!] \Longrightarrow Prop\ C\ \Psi\ P\ P'\ (p \cdot A_P)\ (p \cdot \Psi_P)$
and *rCase*: $\bigwedge \Psi\ P\ P'\ \varphi\ Cs\ A_P\ \Psi_P\ C.\ [\![\Psi \rhd P \longmapsto \tau \prec P'$; $extractFrame\ P = \langle A_P, \Psi_P\rangle$; $distinct\ A_P$; $\bigwedge C.\ Prop\ C\ \Psi\ P\ P'\ A_P\ \Psi_P$;
$(\varphi, P)\ mem\ Cs$; $\Psi \vdash \varphi$; $guarded\ P$; $\Psi_P \simeq \mathbf{1}$; $(supp\ \Psi_P) = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (Cases\ Cs)\ P'\ ([])\ (\mathbf{1})$
and *rPar1*: $\bigwedge \Psi\ \Psi_Q\ P\ P'\ A_Q\ Q\ A_P\ \Psi_P\ C.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto \tau \prec P'$;

$extractFrame\ P = \langle A_P,\ \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_Q)\ P\ P'\ A_P\ \Psi_P$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* P'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* P'$; $A_Q \sharp* \Psi_P$;
$A_P \sharp* C$; $A_Q \sharp* C$$\rrbracket \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (P' \parallel Q)\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rPar2*: $\bigwedge \Psi\ \Psi_P\ Q\ Q'\ A_P\ P\ A_Q\ \Psi_Q\ C.$
$\llbracket \Psi \otimes \Psi_P \rhd Q \longmapsto \tau \prec Q'$;
$extractFrame\ P = \langle A_P,\ \Psi_P\rangle$; $distinct\ A_P$;
$extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$; $distinct\ A_Q$;
$\bigwedge C.\ Prop\ C\ (\Psi \otimes \Psi_P)\ Q\ Q'\ A_Q\ \Psi_Q$;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* \Psi_Q$;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* Q'$; $A_Q \sharp* \Psi_P$;
$A_P \sharp* C$; $A_Q \sharp* C$$\rrbracket \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ (P \parallel Q')\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rComm1*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; $extractFrame\ P = \langle A_P,\ \Psi_P\rangle$; $distinct\ A_P$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; $extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$; $distinct\ A_Q$;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; $distinct\ xvec$;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* K$; $A_Q \sharp* Q'$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$;
$xvec \sharp* Q$; $xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$$\rrbracket \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rComm2*: $\bigwedge \Psi\ \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q\ C.$
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; $extractFrame\ P = \langle A_P,\ \Psi_P\rangle$; $distinct\ A_P$;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; $extractFrame\ Q = \langle A_Q,\ \Psi_Q\rangle$; $distinct\ A_Q$;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; $distinct\ xvec$;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$;
$A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* K$; $A_Q \sharp* Q'$;
$A_Q \sharp* xvec$; $xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$;
$xvec \sharp* Q$; $xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$$\rrbracket \Longrightarrow$
$Prop\ C\ \Psi\ (P \parallel Q)\ ((\!|\nu * xvec|\!)(P' \parallel Q'))\ (A_P @ A_Q)\ (\Psi_P \otimes \Psi_Q)$

and *rScope*: $\bigwedge \Psi\ P\ P'\ x\ A_P\ \Psi_P\ C.$
$\llbracket \Psi \rhd P \longmapsto \tau \prec P'$; $extractFrame\ P = \langle A_P,\ \Psi_P\rangle$; $distinct\ A_P$;
$\bigwedge C.\ Prop\ C\ \Psi\ P\ P'\ A_P\ \Psi_P$; $x \sharp \Psi$;
$x \sharp A_P$; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* P'$;
$A_P \sharp* C$; $x \sharp C$$\rrbracket \Longrightarrow$
$Prop\ C\ \Psi\ ((\!|\nu x|\!)P)\ ((\!|\nu x|\!)P')\ (x \# A_P)\ \Psi_P$

and *rBang*: $\bigwedge \Psi\ P\ P'\ A_P\ \Psi_P\ C.$

$[\![\Psi \rhd P \parallel !P \longmapsto\tau \prec P'$; *guarded* P; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ P'\ A_P\ (\Psi_P \otimes \mathbf{1})$; $\Psi_P \simeq \mathbf{1}$; *supp* $\Psi_P = (\{\}::name\ set)$;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* P'$; $A_P \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ P'\ ([])\ (\mathbf{1})$
shows *Prop* $C\ \Psi\ P\ P'\ A_P\ \Psi_P$
⟨*proof*⟩

lemma *inputFreshDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and $x \sharp P$
and $x \sharp N$

shows $x \sharp P'$
⟨*proof*⟩

lemma *inputFreshChainDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and $xvec \sharp* P$
and $xvec \sharp* N$

shows $xvec \sharp* P'$
⟨*proof*⟩

lemma *outputFreshDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$

and $xvec \sharp* M$
and $distinct\ xvec$
and $x \sharp P$
and $x \sharp xvec$

shows $x \sharp N$
and $x \sharp P'$
⟨*proof*⟩

lemma *outputFreshChainDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $yvec :: name\ list$

assumes $\Psi \rhd P \longmapsto M\langle\!(\nu * xvec)\!\rangle\langle N\rangle \prec P'$
and $xvec \sharp* M$
and $distinct\ xvec$
and $yvec \sharp* P$
and $yvec \sharp* xvec$

shows $yvec \sharp* N$
and $yvec \sharp* P'$
⟨*proof*⟩

lemma *tauFreshDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \longmapsto \tau \prec P'$
and $x \sharp P$

shows $x \sharp P'$
⟨*proof*⟩

lemma *tauFreshChainDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \longmapsto \tau \prec P'$

and $xvec \sharp* P$

shows $xvec \sharp* P'$
⟨*proof*⟩

lemma *freeFreshDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and $bn\ \alpha \sharp* subject\ \alpha$
and $distinct(bn\ \alpha)$
and $x \sharp \alpha$
and $x \sharp P$

shows $x \sharp P'$
⟨*proof*⟩

lemma *freeFreshChainDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and $bn\ \alpha \sharp* subject\ \alpha$
and $distinct(bn\ \alpha)$
and $xvec \sharp* P$
and $xvec \sharp* \alpha$

shows $xvec \sharp* P'$
⟨*proof*⟩

lemma *Input*:
fixes $\Psi :: 'b$
and $M :: 'a$
and $K :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $Tvec :: 'a\ list$

assumes $\Psi \vdash M \leftrightarrow K$
and $distinct\ xvec$
and $set\ xvec \subseteq supp\ N$
and $length\ xvec = length\ Tvec$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto K(\!|N[xvec::=Tvec]|\!) \prec P[xvec::=Tvec]$
⟨*proof*⟩

lemma *residualAlpha*:
fixes $p :: name\ prm$
and $\alpha :: 'a\ action$
and $P :: ('a, 'b, 'c)\ psi$

assumes $bn(p \cdot \alpha) \sharp * object\ \alpha$
and $bn(p \cdot \alpha) \sharp * P$
and $bn\ \alpha \sharp * subject\ \alpha$
and $bn(p \cdot \alpha) \sharp * subject\ \alpha$
and $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$

shows $\alpha \prec P = (p \cdot \alpha) \prec (p \cdot P)$
⟨*proof*⟩

lemma *Par1*:
fixes $\Psi :: 'b$
and $\Psi_Q :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $A_Q :: name\ list$
and $Q :: ('a, 'b, 'c)\ psi$

assumes *Trans*: $\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$
and $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$
and $bn\ \alpha \sharp * Q$
and $A_Q \sharp * \Psi$
and $A_Q \sharp * P$
and $A_Q \sharp * \alpha$

shows $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P' \parallel Q)$
⟨*proof*⟩

lemma *Par2*:
fixes $\Psi :: 'b$
and $\Psi_P :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $Q' :: ('a, 'b, 'c)\ psi$
and $A_P :: name\ list$
and $P :: ('a, 'b, 'c)\ psi$

assumes *Trans*: $\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$
and $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and $bn\ \alpha \sharp * P$

and $A_P \sharp* \Psi$
and $A_P \sharp* Q$
and $A_P \sharp* \alpha$

shows $\Psi \rhd P \parallel Q \longmapsto \alpha \prec (P \parallel Q')$
⟨*proof*⟩

lemma *Open*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and *yvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*

assumes *Trans*: $\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$
and $x \in supp\ N$
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp yvec$

shows $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *Scope*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*

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

shows $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec (\!|\nu x|\!)P'$
⟨*proof*⟩

lemma *inputSwapFrameSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*
and y :: *name*

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and $x \sharp P$
and $y \sharp P$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputPermFrameSubject*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $Xs :: name\ set$
and $Ys :: name\ set$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and S: $set\ p \subseteq Xs \times Ys$
and $Xs \sharp* P$
and $Ys \sharp* P$

shows $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputSwapSubject*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $y :: name$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and $x \sharp P$
and $y \sharp P$
and $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd P \longmapsto ([(x, y)] \cdot M)(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputPermSubject*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $Xs :: name\ set$
and $Ys :: name\ set$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and S: $set\ p \subseteq Xs \times Ys$
and $Xs \sharp* P$
and $Ys \sharp* P$
and $Xs \sharp* \Psi$
and $Ys \sharp* \Psi$

shows $\Psi \rhd P \longmapsto (p \cdot M)(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputSwapFrame*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $y :: name$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and $x \sharp P$
and $y \sharp P$
and $x \sharp M$
and $y \sharp M$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto M(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputPermFrame*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $Xs :: name\ set$
and $Ys :: name\ set$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and S: $set\ p \subseteq Xs \times Ys$
and $Xs \sharp* P$
and $Ys \sharp* P$
and $Xs \sharp* M$
and $Ys \sharp* M$

shows $(p \cdot \Psi) \rhd P \longmapsto M(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *inputAlpha*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*
and *xvec* :: *name list*

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *set p* $\subseteq$ (*set xvec*) $\times$ (*set* $(p \cdot$ *xvec*))
and *distinctPerm p*
and *xvec* $\sharp*$ P
and $(p \cdot$ *xvec*$) \sharp*$ P

shows $\Psi \rhd P \longmapsto M(\!|(p \cdot N)|\!) \prec (p \cdot P')$
⟨*proof*⟩

lemma *frameFresh*[*dest*]:
fixes x :: *name*
and A_F :: *name list*
and Ψ_F :: $'b$

assumes $x \sharp A_F$
and $x \sharp \langle A_F, \Psi_F \rangle$

shows $x \sharp \Psi_F$
⟨*proof*⟩

lemma *outputSwapFrameSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and x :: *name*
and y :: *name*

assumes $\Psi \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N \rangle \prec P'$
and *xvec* $\sharp*$ M
and $x \sharp P$
and $y \sharp P$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto ([(x, y)] \cdot M)(\!|\nu*xvec|\!)\langle N \rangle \prec P'$
⟨*proof*⟩

lemma *outputPermFrameSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*
and *yvec* :: *name list*
and *zvec* :: *name list*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and S: *set* $p \subseteq$ *set yvec* $\times$ *set zvec*
and *yvec* $\sharp*$ P
and *zvec* $\sharp*$ P

shows $(p \cdot \Psi) \rhd P \longmapsto (p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *outputSwapSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and x :: *name*
and y :: *name*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and *xvec* $\sharp*$ M
and $x \sharp P$
and $y \sharp P$
and $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd P \longmapsto ([(x, y)] \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *outputPermSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and p :: *name prm*
and *yvec* :: *name list*
and *zvec* :: *name list*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and S: *set* $p \subseteq$ *set yvec* $\times$ *set zvec*

and $yvec \sharp* P$
and $zvec \sharp* P$
and $yvec \sharp* \Psi$
and $zvec \sharp* \Psi$

shows $\Psi \rhd P \longmapsto (p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *outputSwapFrame*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and x :: *name*
and y :: *name*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and $xvec \sharp* M$
and $x \sharp P$
and $y \sharp P$
and $x \sharp M$
and $y \sharp M$

shows $([(x, y)] \cdot \Psi) \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *outputPermFrame*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and B :: $('a, 'b, 'c)$ *boundOutput*
and p :: *name prm*
and *yvec* :: *name list*
and *zvec* :: *name list*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and S: *set p* $\subseteq$ *set yvec* $\times$ *set zvec*
and $yvec \sharp* P$
and $zvec \sharp* P$
and $yvec \sharp* M$
and $zvec \sharp* M$

shows $(p \cdot \Psi) \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *Comm1*:
fixes Ψ :: $'b$
and Ψ_Q :: $'b$
and P :: $('a, 'b, 'c)$ *psi*

and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and K :: $'a$
and *xvec* :: *name list*
and Q' :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*

assumes $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$
and $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
and $A_P \sharp* \Psi$
and $A_P \sharp* P$
and $A_P \sharp* Q$
and $A_P \sharp* M$
and $A_P \sharp* A_Q$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$
and $A_Q \sharp* Q$
and $A_Q \sharp* K$
and *xvec* $\sharp*$ P

shows $\Psi \rhd P \parallel Q \longmapsto \tau \prec (\!|\nu * xvec|\!)(P' \parallel Q')$
⟨*proof*⟩

lemma *Comm2*:
fixes Ψ :: $'b$
and Ψ_Q :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and K :: $'a$
and Q' :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*

assumes $\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and $\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q \rangle$

and $\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$
and $A_P \sharp* \Psi$
and $A_P \sharp* P$
and $A_P \sharp* Q$
and $A_P \sharp* M$
and $A_P \sharp* A_Q$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$
and $A_Q \sharp* Q$
and $A_Q \sharp* K$
and $xvec \sharp* Q$

shows $\Psi \rhd P \parallel Q \longmapsto\tau \prec (\!|\nu*xvec|\!)(P' \parallel Q')$
⟨*proof*⟩

lemma *semanticsCasesAux*[*consumes 1*, *case-names cInput cOutput cCase cPar1 cPar2 cComm1 cComm2 cOpen cScope cBang*]:
fixes Ψ :: *'b*
and *cP* :: (*'a*, *'b*, *'c*) *psi*
and *cRs* :: (*'a*, *'b*, *'c*) *residual*
and *C* :: *'d*::*fs-name*
and *x* :: *name*

assumes $\Psi \rhd cP \longmapsto cRs$
and *rInput*: $\bigwedge M\ K\ xvec\ N\ Tvec\ P.\ [\![cP = M(\!|\lambda*xvec\ N|\!).P;\ cRs = K(\!|(N[xvec::=Tvec])|\!) \prec P[xvec::=Tvec];$
$\Psi \vdash M \leftrightarrow K$; *distinct xvec*; *set xvec* $\subseteq$ *supp N*; *length xvec*=*length Tvec*;
xvec $\sharp*$ *Tvec*; *xvec* $\sharp*$ Ψ; *xvec* $\sharp*$ *M*; *xvec* $\sharp*$ *K*; *xvec* $\sharp*$ *C*$]\!] \Longrightarrow$ *Prop*
and *rOutput*: $\bigwedge M\ K\ N\ P.\ [\![cP = M\langle N\rangle.P;\ cRs = K\langle N\rangle \prec P;\ \Psi \vdash M \leftrightarrow K]\!] \Longrightarrow$ *Prop*
and *rCase*: $\bigwedge Cs\ P\ \varphi.\ [\![cP = Cases\ Cs;\ \Psi \rhd P \longmapsto cRs;\ (\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow$ *Prop*

and *rPar1*: $\bigwedge \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q.\ [\![cP = P \parallel Q;\ cRs = \alpha \prec (P' \parallel Q);$
$(\Psi \otimes \Psi_Q) \rhd P \longmapsto (\alpha \prec P')$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \Psi$; $A_Q \sharp* \alpha$; $A_Q \sharp* C$; $A_Q \sharp* P'$; *bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* \Psi_Q$;
bn $\alpha \sharp* Q$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp*$ *subject* α; *bn* $\alpha \sharp* C$; *distinct*(*bn* α)$]\!] \Longrightarrow$
Prop
and *rPar2*: $\bigwedge \Psi_P\ Q\ \alpha\ Q'\ P\ A_P.\ [\![cP = P \parallel Q;\ cRs = \alpha \prec (P \parallel Q');$
$(\Psi \otimes \Psi_P) \rhd Q \longmapsto \alpha \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \Psi$; $A_P \sharp* \alpha$; $A_P \sharp* C$;
$A_P \sharp* Q'$; *bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* \Psi_P$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp* Q$; *bn* $\alpha \sharp*$ *subject* α; *bn* $\alpha \sharp* C$; *distinct*(*bn* α)$]\!] \Longrightarrow$ *Prop*

and *rComm1*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q.$
$[\![cP = P \parallel Q;\ cRs = \tau \prec (\!|\nu * xvec|\!) P' \parallel Q';$
$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;$
$distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;$
$A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_Q \sharp* xvec;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* Q;$
$xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop$
and *rComm2*: $\bigwedge \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q.$
$[\![cP = P \parallel Q;\ cRs = \tau \prec (\!|\nu * xvec|\!) P' \parallel Q';$
$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N \rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P \rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q \rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;$
$A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_Q \sharp* xvec;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* Q;$
$xvec \sharp* K;\ A_P \sharp* C;\ A_Q \sharp* C;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop$
and *rOpen*: $\bigwedge P\ M\ xvec\ yvec\ N\ P'\ x.$
$[\![cP = (\!|\nu x|\!) P;\ cRs = M(\!|\nu * (xvec @ x \# yvec)|\!)\langle N \rangle \prec P';$
$\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N \rangle \prec P';\ x \in supp\ N;\ x \sharp xvec;\ x \sharp yvec;\ x \sharp M;\ x \sharp \Psi;\ distinct\ xvec;\ distinct\ yvec;$
$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* yvec;\ yvec \sharp* \Psi;\ yvec \sharp* P;\ yvec \sharp* M;\ xvec \sharp* C;\ x \sharp C;\ yvec \sharp* C]\!] \Longrightarrow$
$Prop$
and *rScope*: $\bigwedge P\ \alpha\ P'\ x.\ [\![cP = (\!|\nu x|\!) P;\ cRs = \alpha \prec (\!|\nu x|\!) P';$
$\Psi \rhd P \longmapsto \alpha \prec P';\ x \sharp \Psi;\ x \sharp \alpha;\ x \sharp C;\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ distinct(bn\ \alpha)]\!] \Longrightarrow Prop$
and *rBang*: $\bigwedge P.\ [\![cP = !P;\ \Psi \rhd P \parallel !P \longmapsto cRs;\ guarded\ P]\!] \Longrightarrow Prop$
shows *Prop*
⟨*proof*⟩

nominal-primrec
inputLength :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psi* ⇒ *nat*
and *inputLength'* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *input* ⇒ *nat*
and *inputLength''* :: ('*a*::*fs-name*, '*b*::*fs-name*, '*c*::*fs-name*) *psiCase* ⇒ *nat*

where
  inputLength (**0**) = *0*
| *inputLength* (*M*⟨*N*⟩.*P*) = *0*
| *inputLength* (*M*(⦇*I*) = *inputLength′ I*
| *inputLength* (*Case C*) = *0*
| *inputLength* (*P* ∥ *Q*) = *0*
| *inputLength* (⦇*νx*⦈*P*) = *0*
| *inputLength* (⦃Ψ⦄) = *0*
| *inputLength* (!*P*) = *0*

| *inputLength′* (*Trm M P*) = *0*
| *inputLength′* (*ν y I*) = *1* + (*inputLength′ I*)

| *inputLength″* (⊥$_c$) = *0*
| *inputLength″* (□Φ ⇒ *P C*) = *0*
⟨*proof*⟩

nominal-primrec *boundOutputLength* :: (′*a*, ′*b*, ′*c*) *boundOutput* ⇒ *nat*
where
  boundOutputLength (*BOut M P*) = *0*
| *boundOutputLength* (*BStep x B*) = (*boundOutputLength B*) + *1*
⟨*proof*⟩

nominal-primrec *residualLength* :: (′*a*, ′*b*, ′*c*) *residual* ⇒ *nat*
where
  residualLength (*RIn M N P*) = *0*
| *residualLength* (*ROut M B*) = *boundOutputLength B*
| *residualLength* (*RTau P*) = *0*
⟨*proof*⟩

lemma *inputLengthProc*[*simp*]:
  shows *inputLength*(*M*(⦇λ∗*xvec N*⦈).*P*) = *length xvec*
⟨*proof*⟩

lemma *boundOutputLengthSimp*[*simp*]:
  shows *residualLength*(*M*(⦇*ν*∗*xvec*⦈)⟨*N*⟩ ≺ *P*) = *length xvec*
⟨*proof*⟩

lemma *boundOuputLengthSimp2*[*simp*]:
  shows *residualLength*(*α* ≺ *P*) = *length*(*bn α*)
⟨*proof*⟩

lemmas [*simp del*] = *inputLength-inputLength′-inputLength″.simps residualLength.simps boundOutputLength.simps*

lemma *constructPerm*:
  fixes *xvec* :: *name list*
  and *yvec* :: *name list*

assumes *length xvec* = *length yvec*
and *xvec* ♯* *yvec*
and *distinct xvec*
and *distinct yvec*

obtains *p* **where** *set p* ⊆ *set xvec* × *set*(*p* ∙ *xvec*) **and** *distinctPerm p* **and** *yvec* = *p* ∙ *xvec*
⟨*proof*⟩

lemma *distinctApend*[*simp*]:
fixes *xvec* :: *name list*
and *yvec* :: *name list*

shows (*set xvec* ∩ *set yvec* = {}) = *xvec* ♯* *yvec*
⟨*proof*⟩

lemma *lengthAux*:
fixes *xvec* :: *name list*
and *y* :: *name*
and *yvec* :: *name list*

assumes *length xvec* = *length*(*y*#*yvec*)

obtains *z zvec* **where** *xvec* = *z*#*zvec* **and** *length zvec* = *length yvec*
⟨*proof*⟩

lemma *lengthAux2*:
fixes *xvec* :: *name list*
and *yvec* :: *name list*
and *zvec* :: *name list*

assumes *length xvec* = *length*(*yvec*@*y*#*zvec*)

obtains *xvec1 x xvec2* **where** *xvec*=*xvec1*@*x*#*xvec2* **and** *length xvec1* = *length yvec* **and** *length xvec2* = *length zvec*
⟨*proof*⟩

lemma *semanticsCases*[*consumes 11*, *case-names cInput cCase cPar1 cPar2 cComm1 cComm2 cScope cBang*]:
fixes Ψ :: *'b*
and *cP* :: (*'a*, *'b*, *'c*) *psi*
and *cRs* :: (*'a*, *'b*, *'c*) *residual*
and *C* :: *'d*::*fs-name*
and *x1* :: *name*
and *x2* :: *name*
and *xvec1* :: *name list*
and *xvec2* :: *name list*
and *xvec3* :: *name list*

and *xvec4* :: *name list*
and *xvec5* :: *name list*

assumes $\Psi \rhd cP \longmapsto cRs$
and *length xvec1* = *inputLength cP* **and** *distinct xvec1*
and *length xvec2* = *residualLength cRs* **and** *distinct xvec2*
and *length xvec3* = *residualLength cRs* **and** *distinct xvec3*
and *length xvec4* = *residualLength cRs* **and** *distinct xvec4*
and *length xvec5* = *residualLength cRs* **and** *distinct xvec5*
and *rInput*: $\bigwedge M\ K\ N\ Tvec\ P.\ ([\![xvec1 \sharp* \Psi;\ xvec1 \sharp* cP;\ xvec1 \sharp* cRs]\!] \Longrightarrow cP = M(\!|\lambda* xvec1\ N|\!).P \wedge\ \ cRs = K(\!|(N[xvec1::=Tvec])|\!) \prec P[xvec1::=Tvec] \wedge \Psi \vdash M \leftrightarrow K \wedge distinct\ xvec1 \wedge set\ xvec1 \subseteq supp\ N \wedge length\ xvec1 = length\ Tvec \wedge xvec1 \sharp* Tvec \wedge xvec1 \sharp* \Psi \wedge xvec1 \sharp* M \wedge xvec1 \sharp* K) \Longrightarrow Prop$
and *rOutput*: $\bigwedge M\ K\ N\ P.\ [\![cP = M\langle N\rangle.P;\ cRs = K\langle N\rangle \prec P;\ \Psi \vdash M \leftrightarrow K]\!] \Longrightarrow Prop$
and *rCase*: $\bigwedge Cs\ P\ \varphi.\ [\![cP = Cases\ Cs;\ \Psi \rhd P \longmapsto cRs;\ (\varphi, P)\ mem\ Cs;\ \Psi \vdash \varphi;\ guarded\ P]\!] \Longrightarrow Prop$
and *rPar1*: $\bigwedge \Psi_Q\ P\ \alpha\ P'\ Q\ A_Q.\ ([\![xvec2 \sharp* \Psi;\ xvec2 \sharp* cP;\ xvec2 \sharp* cRs]\!] \Longrightarrow cP = P \parallel Q \wedge cRs = \alpha \prec (P' \parallel Q) \wedge xvec2 = bn\ \alpha \wedge \Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P' \wedge extractFrame\ Q = \langle A_Q, \Psi_Q\rangle \wedge distinct\ A_Q \wedge A_Q \sharp* P \wedge A_Q \sharp* Q \wedge A_Q \sharp* \Psi \wedge A_Q \sharp* \alpha \wedge A_Q \sharp* P' \wedge A_Q \sharp* C) \Longrightarrow Prop$
and *rPar2*: $\bigwedge \Psi_P\ Q\ \alpha\ Q'\ P\ A_P.\ ([\![xvec3 \sharp* \Psi;\ xvec3 \sharp* cP;\ xvec3 \sharp* cRs]\!] \Longrightarrow cP = P \parallel Q \wedge cRs = \alpha \prec (P \parallel Q') \wedge xvec3 = bn\ \alpha \wedge \Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q' \wedge extractFrame\ P = \langle A_P, \Psi_P\rangle \wedge distinct\ A_P \wedge A_P \sharp* P \wedge A_P \sharp* Q \wedge A_P \sharp* \Psi \wedge A_P \sharp* \alpha \wedge A_P \sharp* Q' \wedge A_P \sharp* C) \Longrightarrow Prop$
and *rComm1*: $\bigwedge \Psi_Q\ P\ M\ N\ P'\ A_P\ \Psi_P\ Q\ K\ xvec\ Q'\ A_Q.$
$[\![cP = P \parallel Q;\ cRs = \tau \prec (\!|\nu* xvec|\!)P' \parallel Q';$
$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;$
$A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* xvec;\ A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;$
$A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* Q';\ A_Q \sharp* xvec;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* \Psi_Q;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* Q;$

$xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$; *distinct xvec*$\rrbracket \Longrightarrow$ *Prop*
and *rComm2*: $\bigwedge \Psi_Q\ P\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ Q\ K\ Q'\ A_Q$.
$\llbracket cP = P \parallel Q$; $cRs = \tau \prec (\!|\nu * xvec|\!) P' \parallel Q'$;
$\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; $A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$;
$A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* xvec$; $A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$;
$A_Q \sharp* P$; $A_Q \sharp* K$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* Q'$; $A_Q \sharp* xvec$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* \Psi_Q$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* Q$;
$xvec \sharp* K$; $A_P \sharp* C$; $A_Q \sharp* C$; $xvec \sharp* C$; *distinct xvec*$\rrbracket \Longrightarrow$ *Prop*
and *rOpen*: $\bigwedge P\ M\ xvec\ y\ yvec\ N\ P'$.
($\llbracket xvec4 \sharp* \Psi$; $xvec4 \sharp* cP$; $xvec4 \sharp* cRs$; $x1 \sharp \Psi$; $x1 \sharp cP$; $x1 \sharp cRs$; $x1 \sharp xvec4\rrbracket \Longrightarrow$
$cP = (\!|\nu x1|\!) P \wedge cRs = M(\!|\nu * (xvec @ x1 \# yvec)|\!)\langle N\rangle \prec P' \wedge xvec4 = xvec @ y \# yvec \wedge$
$\Psi \rhd P \longmapsto M(\!|\nu * (xvec @ yvec)|\!)\langle N\rangle \prec P' \wedge x1 \in supp\ N \wedge x1 \sharp xvec \wedge x1 \sharp yvec \wedge$
$distinct\ xvec \wedge distinct\ yvec \wedge xvec \sharp* \Psi \wedge xvec \sharp* P \wedge xvec \sharp* M \wedge xvec \sharp* yvec \wedge$
$yvec \sharp* \Psi \wedge yvec \sharp* P \wedge yvec \sharp* M) \Longrightarrow$ *Prop*
and *rScope*: $\bigwedge P\ \alpha\ P'$. ($\llbracket xvec5 \sharp* \Psi$; $xvec5 \sharp* cP$; $xvec5 \sharp* cRs$; $x2 \sharp \Psi$; $x2 \sharp cP$; $x2 \sharp cRs$; $x2 \sharp xvec5\rrbracket \Longrightarrow$
$cP = (\!|\nu x2|\!) P \wedge cRs = \alpha \prec (\!|\nu x2|\!) P' \wedge\ xvec5 = bn\ \alpha \wedge$
$\Psi \rhd P \longmapsto \alpha \prec P' \wedge x2 \sharp \Psi \wedge x2 \sharp \alpha \wedge bn\ \alpha \sharp* subject\ \alpha \wedge distinct(bn\ \alpha)) \Longrightarrow$ *Prop*
and *rBang*: $\bigwedge P$. $\llbracket cP = !P$; $\Psi \rhd P \parallel !P \longmapsto cRs$; *guarded* $P\rrbracket \Longrightarrow$ *Prop*
shows *Prop*
⟨*proof*⟩

lemma *parCases*[*consumes 5*, *case-names cPar1 cPar2 cComm1 cComm2*]:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and Q :: *('a, 'b, 'c) psi*
and α :: *'a action*
and T :: *('a, 'b, 'c) psi*
and C :: *'d::fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto \alpha \prec T$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* P$
and $bn\ \alpha \sharp* Q$
and $bn\ \alpha \sharp* subject\ \alpha$
and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q$. $\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P'$; *extractFrame* $Q = \langle A_Q,$

$\Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* \alpha$; $A_Q \sharp* P'$; $A_Q \sharp* C]\!] \Longrightarrow$ *Prop* α $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P$. $[\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* \alpha$; $A_P \sharp* Q'$; $A_P \sharp* C]\!] \Longrightarrow$ *Prop* α $(P \parallel Q')$
and *rComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *distinct xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* K$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* K$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$; $xvec \sharp* C]\!] \Longrightarrow$
Prop (τ) $((\!|\nu * xvec|\!)(P' \parallel Q'))$
and *rComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ K\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *distinct xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* K$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* K$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$; $xvec \sharp* C]\!] \Longrightarrow$
Prop (τ) $((\!|\nu * xvec|\!)(P' \parallel Q'))$

shows *Prop* α T
⟨*proof*⟩

lemma *parInputCases*[*consumes 1*, *case-names cPar1 cPar2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)$ *psi*
and $Q :: ('a, 'b, 'c)$ *psi*
and $M :: 'a$
and $N :: 'a$
and $R :: ('a, 'b, 'c)$ *psi*
and $C :: 'd$::*fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto M(\!|N|\!) \prec R$
and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q$. $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$; $A_Q \sharp* N$; $A_Q \sharp* C]\!] \Longrightarrow$ *Prop* $(P' \parallel Q)$

and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* C]\!] \Longrightarrow Prop\ (P \parallel Q')$
shows *Prop R*
⟨*proof*⟩

lemma *parOutputCases*[*consumes 5*, *case-names cPar1 cPar2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $R :: ('a, 'b, 'c)\ psi$
and $C :: 'd::fs\text{-}name$

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec R$
and $xvec \sharp* \Psi$
and $xvec \sharp* P$
and $xvec \sharp* Q$
and $xvec \sharp* M$
and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_Q \sharp* \Psi;\ A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* M;\ A_Q \sharp* xvec;\ A_Q \sharp* N;\ A_Q \sharp* C;\ A_Q \sharp* xvec;\ distinct\ xvec]\!] \Longrightarrow Prop\ (P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* M;\ A_P \sharp* xvec;\ A_P \sharp* N;\ A_P \sharp* C;\ A_P \sharp* xvec;\ distinct\ xvec]\!] \Longrightarrow Prop\ (P \parallel Q')$
shows *Prop R*
⟨*proof*⟩

lemma *theEqvt*[*eqvt-force*]:
fixes $p :: name\ prm$
and $\alpha :: 'a\ action$

assumes $\alpha \neq \tau$

shows $(p \cdot the(subject\ \alpha)) = the(p \cdot (subject\ \alpha))$
⟨*proof*⟩

lemma *theSubjectFresh*[*simp*]:
fixes $\alpha :: 'a\ action$
and $x :: name$

assumes $\alpha \neq \tau$

shows $x \sharp the(subject\ \alpha) = x \sharp subject\ \alpha$

⟨*proof*⟩

lemma *theSubjectFreshChain*[*simp*]:
fixes α :: $'a$ *action*
and *xvec* :: *name list*

assumes $\alpha \neq \tau$

shows *xvec* $\sharp*$ *the*(*subject* α) = *xvec* $\sharp*$ *subject* α
⟨*proof*⟩

lemma *obtainPrefix*:
fixes Ψ :: $'b$
and P :: ($'a$, $'b$, $'c$) *psi*
and α :: $'a$ *action*
and P' :: ($'a$, $'b$, $'c$) *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and B :: *name list*

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and *bn* $\alpha \sharp*$ *subject* α
and *distinct*(*bn* α)
and $\alpha \neq \tau$
and $B \sharp* P$
and $A_P \sharp* \Psi$
and $A_P \sharp* B$
and $A_P \sharp* P$
and $A_P \sharp*$ *subject* α

obtains M **where** $\Psi \otimes \Psi_P \vdash$ *the*(*subject* α) $\leftrightarrow M$ **and** $B \sharp* M$
⟨*proof*⟩

lemma *inputRenameSubject*:
fixes Ψ :: $'b$
and P :: ($'a$, $'b$, $'c$) *psi*
and M :: $'a$
and N :: $'a$
and P' :: ($'a$, $'b$, $'c$) *psi*
and A_P :: *name list*
and Ψ_P :: $'b$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$
and $A_P \sharp* \Psi$

and $A_P \sharp* P$
and $A_P \sharp* M$
and $A_P \sharp* K$

shows $\Psi \rhd P \longmapsto K(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *outputRenameSubject*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$

assumes $\Psi \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *distinct* A_P
and $\Psi \otimes \Psi_P \vdash M \leftrightarrow K$
and $A_P \sharp* \Psi$
and $A_P \sharp* P$
and $A_P \sharp* M$
and $A_P \sharp* K$

shows $\Psi \rhd P \longmapsto K(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *parCasesSubject*[*consumes 7*, *case-names cPar1 cPar2 cComm1 cComm2*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and R :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*
and *yvec* :: *name list*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto \alpha \prec R$
and *bn* $\alpha \sharp* \Psi$
and *bn* $\alpha \sharp* P$
and *bn* $\alpha \sharp* Q$
and *bn* $\alpha \sharp*$ *subject* α
and *yvec* $\sharp* P$
and *yvec* $\sharp* Q$
and *rPar1*: $\bigwedge P'\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto \alpha \prec P';$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* \alpha$; $A_Q \sharp* C]\!] \Longrightarrow$ *Prop* α $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto \alpha \prec Q';$ *extractFrame* $P = \langle A_P,$

$\Psi_P\rangle$; *distinct* A_P;
$A_P \sharp^* \Psi$; $A_P \sharp^* Q$; $A_P \sharp^* \alpha$; $A_P \sharp^* C\rrbracket \Longrightarrow$ *Prop* α $(P \parallel Q')$
and *rComm1*: $\bigwedge \Psi_Q$ M N P' A_P Ψ_P K *xvec* Q' A_Q.
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *yvec* $\sharp^* M$; *yvec* $\sharp^* K$; *distinct xvec*;
$A_P \sharp^* \Psi$; $A_P \sharp^* \Psi_Q$; $A_P \sharp^* P$; $A_P \sharp^* M$; $A_P \sharp^* N$; $A_P \sharp^* P'$; $A_P \sharp^* Q$; $A_P \sharp^*$ *xvec*; $A_P \sharp^* Q'$; $A_P \sharp^* A_Q$; $A_P \sharp^* C$;
$A_Q \sharp^* \Psi$; $A_Q \sharp^* \Psi_P$; $A_Q \sharp^* P$; $A_Q \sharp^* K$; $A_Q \sharp^* N$; $A_Q \sharp^* P'$; $A_Q \sharp^* Q$; $A_Q \sharp^*$ *xvec*; $A_Q \sharp^* Q'$; $A_Q \sharp^* C$;
xvec $\sharp^* \Psi$; *xvec* $\sharp^* \Psi_P$; *xvec* $\sharp^* P$; *xvec* $\sharp^* M$; *xvec* $\sharp^* K$; *xvec* $\sharp^* Q$; *xvec* $\sharp^* \Psi_Q$; *xvec* $\sharp^* C\rrbracket \Longrightarrow$
Prop (τ) $((\!|\nu * xvec|\!)(P' \parallel Q'))$
and *rComm2*: $\bigwedge \Psi_Q$ M *xvec* N P' A_P Ψ_P K Q' A_Q.
$\llbracket \Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *yvec* $\sharp^* M$; *yvec* $\sharp^* K$; *distinct xvec*;
$A_P \sharp^* \Psi$; $A_P \sharp^* \Psi_Q$; $A_P \sharp^* P$; $A_P \sharp^* M$; $A_P \sharp^* N$; $A_P \sharp^* P'$; $A_P \sharp^* Q$; $A_P \sharp^*$ *xvec*; $A_P \sharp^* Q'$; $A_P \sharp^* A_Q$; $A_P \sharp^* C$;
$A_Q \sharp^* \Psi$; $A_Q \sharp^* \Psi_P$; $A_Q \sharp^* P$; $A_Q \sharp^* K$; $A_Q \sharp^* N$; $A_Q \sharp^* P'$; $A_Q \sharp^* Q$; $A_Q \sharp^*$ *xvec*; $A_Q \sharp^* Q'$; $A_Q \sharp^* C$;
xvec $\sharp^* \Psi$; *xvec* $\sharp^* \Psi_P$; *xvec* $\sharp^* P$; *xvec* $\sharp^* M$; *xvec* $\sharp^* K$; *xvec* $\sharp^* Q$; *xvec* $\sharp^* \Psi_Q$; *xvec* $\sharp^* C\rrbracket \Longrightarrow$
Prop (τ) $((\!|\nu * xvec|\!)(P' \parallel Q'))$

shows *Prop* α R
⟨*proof*⟩

lemma *inputCases*[*consumes 1*, *case-names cInput*]:
fixes Ψ :: *'b*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and P :: *('a, 'b, 'c) psi*
and α :: *'a action*
and P' :: *('a, 'b, 'c) psi*

assumes *Trans*: $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P'$
and *rInput*: $\bigwedge K$ *Tvec*. $\llbracket \Psi \vdash M \leftrightarrow K$; *set xvec* $\subseteq$ *supp* N; *length xvec* = *length Tvec*; *distinct xvec*$\rrbracket \Longrightarrow$ *Prop* $(K(\!|N[xvec::=Tvec]|\!))$ $(P[xvec::=Tvec])$

shows *Prop* α P'
⟨*proof*⟩

lemma *outputCases*[*consumes 1*, *case-names cOutput*]:
fixes Ψ :: *'b*
and M :: *'a*

and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd M\langle N\rangle.P \longmapsto \alpha \prec P'$
and $\bigwedge K.\ \Psi \vdash M \leftrightarrow K \Longrightarrow Prop\ (K\langle N\rangle)\ P$

shows $Prop\ \alpha\ P'$
⟨*proof*⟩

lemma *caseCases*[*consumes 1*, *case-names cCase*]:
fixes $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *Trans*: $\Psi \rhd (Cases\ Cs) \longmapsto Rs$
and *rCase*: $\bigwedge \varphi\ P.\ [\![\Psi \rhd P \longmapsto Rs; (\varphi, P)\ mem\ Cs; \Psi \vdash \varphi; guarded\ P]\!] \Longrightarrow Prop$

shows *Prop*
⟨*proof*⟩

lemma *resCases*[*consumes 7*, *case-names cOpen cRes*]:
fixes $\Psi :: 'b$
and $x :: name$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $C :: 'd::fs\text{-}name$

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec P'$
and $x \sharp \Psi$
and $x \sharp \alpha$
and $x \sharp P'$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* P$
and $bn\ \alpha \sharp* subject\ \alpha$
and *rOpen*: $\bigwedge M\ xvec\ yvec\ y\ N\ P'.\ [\![\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot P'); y \in supp\ N;$
$x \sharp N; x \sharp P'; x \neq y; y \sharp xvec; y \sharp yvec; y \sharp M;$
$distinct\ xvec; distinct\ yvec;$
$xvec \sharp* \Psi; y \sharp \Psi; yvec \sharp* \Psi; xvec \sharp* P; y \sharp P;$
$yvec \sharp* P; xvec \sharp* M; y \sharp M;$
$yvec \sharp* M; xvec \sharp* yvec]\!] \Longrightarrow$
$Prop\ (M(\!|\nu*(xvec@y\#yvec)|\!)\langle N\rangle)\ P'$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto \alpha \prec P']\!] \Longrightarrow Prop\ \alpha\ ((\!|\nu x|\!)P')$

shows *Prop* α P'
⟨*proof*⟩

lemma *resCases'*[*consumes* 7, *case-names cOpen cRes*]:
fixes Ψ :: $'b$
and x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \alpha \prec P'$
and $x \sharp \Psi$
and $x \sharp \alpha$
and $x \sharp P'$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* P$
and $bn\ \alpha \sharp* subject\ \alpha$
and *rOpen*: $\bigwedge M\ xvec\ yvec\ y\ N\ P'.\ [\![\Psi \rhd ([(x, y)] \cdot P) \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P';\ y \in supp\ N;$
$x \sharp N;\ x \sharp P';\ x \neq y;\ y \sharp xvec;\ y \sharp yvec;\ y \sharp M;$ *distinct xvec*; *distinct yvec*;
$xvec \sharp* \Psi;\ y \sharp \Psi;\ yvec \sharp* \Psi;\ xvec \sharp* P;\ y \sharp P;$ $yvec \sharp* P;\ xvec \sharp* M;\ y \sharp M;$
$yvec \sharp* M;\ xvec \sharp* yvec]\!] \Longrightarrow$
Prop $(M(\!|\nu*(xvec@y\#yvec)|\!)\langle N\rangle)\ P'$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto \alpha \prec P']\!] \Longrightarrow Prop\ \alpha\ ((\!|\nu x|\!)P')$

shows *Prop* α P'
⟨*proof*⟩

abbreviation
statImpJudge (‹- ↪ -› [80, 80] 80)
where $\Psi \hookrightarrow \Psi' \equiv AssertionStatImp\ \Psi\ \Psi'$

lemma *statEqTransition*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Rs :: $('a, 'b, 'c)$ *residual*
and Ψ' :: $'b$

assumes $\Psi \rhd P \longmapsto Rs$
and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \longmapsto Rs$
⟨*proof*⟩

lemma *actionPar1Dest'*:
fixes α :: $('a$::*fs-name*$)$ *action*

and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $\beta :: ('a::fs\text{-}name)\ action$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $bn\ \alpha \sharp* R$
and $bn\ \beta \sharp* R$

obtains T **where** $P = T \parallel R$ **and** $\alpha \prec T = \beta \prec Q$
⟨*proof*⟩

lemma *actionPar2Dest'*:
fixes $\alpha :: ('a::fs\text{-}name)\ action$
and $P :: ('a, 'b::fs\text{-}name, 'c::fs\text{-}name)\ psi$
and $\beta :: ('a::fs\text{-}name)\ action$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\alpha \prec P = \beta \prec (Q \parallel R)$
and $bn\ \alpha \sharp* Q$
and $bn\ \beta \sharp* Q$

obtains T **where** $P = Q \parallel T$ **and** $\alpha \prec T = \beta \prec R$
⟨*proof*⟩

lemma *expandNonTauFrame*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $A_P :: name\ list$
and $\Psi_P :: 'b$
and $C :: 'd::fs\text{-}name$
and $C' :: 'e::fs\text{-}name$

assumes *Trans*: $\Psi \rhd P \longmapsto \alpha \prec P'$
and $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and $distinct\ A_P$
and $bn\ \alpha \sharp* subject\ \alpha$
and $A_P \sharp* P$
and $A_P \sharp* \alpha$
and $A_P \sharp* C$
and $A_P \sharp* C'$
and $bn\ \alpha \sharp* P$
and $bn\ \alpha \sharp* C'$
and $\alpha \neq \tau$

obtains $p\ \Psi'\ A_P'\ \Psi_P'$ **where** $set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **and** $(p \cdot \Psi_P)$

$\otimes\ \Psi' \simeq \Psi_P'$ **and** *distinctPerm* p **and**
extractFrame $P' = \langle A_P', \Psi_P' \rangle$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* \alpha$ **and** $A_P' \sharp* (p \cdot \alpha)$ **and**
$A_P' \sharp* C$ **and** $(bn(p \cdot \alpha)) \sharp* C'$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$ **and** $(bn(p \cdot \alpha)) \sharp* P'$ **and** *distinct* A_P'
⟨*proof*⟩

lemma *expandTauFrame*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and C :: $'d$::*fs-name*

assumes $\Psi \rhd P \longmapsto \tau \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $A_P \sharp* P$
and $A_P \sharp* C$

obtains $\Psi'\ A_P'\ \Psi_P'$ **where** *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and** $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** $A_P' \sharp* C$ **and** $A_P' \sharp* P'$ **and** *distinct* A_P'
⟨*proof*⟩

lemma *expandFrame*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and C :: $'d$::*fs-name*
and C' :: $'e$::*fs-name*

assumes *Trans*: $\Psi \rhd P \longmapsto \alpha \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and *bn* $\alpha \sharp* subject\ \alpha$
and *distinct*(*bn* α)
and $A_P \sharp* \alpha$
and $A_P \sharp* P$
and $A_P \sharp* C$
and $A_P \sharp* C'$
and *bn* $\alpha \sharp* P$
and *bn* $\alpha \sharp* C'$

obtains $p\ \Psi'\ A_P'\ \Psi_P'$ **where** *set* $p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha))$ **and** $(p \cdot \Psi_P) \otimes \Psi' \simeq \Psi_P'$ **and** *distinctPerm* p **and**

extractFrame $P' = \langle A_P', \Psi_P' \rangle$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* \alpha$ **and** $A_P' \sharp* (p \cdot \alpha)$ **and**
$A_P' \sharp* C$ **and** $(bn(p \cdot \alpha)) \sharp* C'$ **and** $(bn(p \cdot \alpha)) \sharp* \alpha$ **and** $(bn(p \cdot \alpha)) \sharp* P'$ **and** *distinct* A_P'
⟨*proof*⟩

abbreviation
frameImpJudge (‹- $\hookrightarrow_F$ -› [80, 80] 80)
where $F \hookrightarrow_F G \equiv$ *FrameStatImp* F G

lemma *FrameStatEqImpCompose*:
fixes F :: 'b *frame*
and G :: 'b *frame*
and H :: 'b *frame*
and I :: 'b *frame*

assumes $F \simeq_F G$
and $G \hookrightarrow_F H$
and $H \simeq_F I$

shows $F \hookrightarrow_F I$
⟨*proof*⟩

lemma *transferNonTauFrame*:
fixes Ψ_F :: 'b
and P :: ('a, 'b, 'c) *psi*
and α :: 'a *action*
and P' :: ('a, 'b, 'c) *psi*
and A_F :: *name list*
and A_G :: *name list*
and Ψ_G :: 'b

assumes $\Psi_F \rhd P \longmapsto \alpha \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and *distinct*(*bn* α)
and $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
and $A_F \sharp* P$
and $A_G \sharp* P$
and $A_F \sharp* subject\ \alpha$
and $A_G \sharp* subject\ \alpha$
and $A_P \sharp* A_F$
and $A_P \sharp* A_G$
and $A_P \sharp* \Psi_F$
and $A_P \sharp* \Psi_G$
and $\alpha \neq \tau$

shows $\Psi_G \rhd P \longmapsto \alpha \prec P'$
⟨*proof*⟩

lemma *transferTauFrame*:
fixes Ψ_F :: *'b*
and P :: *('a, 'b, 'c) psi*
and P' :: *('a, 'b, 'c) psi*
and A_F :: *name list*
and A_G :: *name list*
and Ψ_G :: *'b*

assumes $\Psi_F \rhd P \longmapsto \tau \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
and $A_F \sharp* P$
and $A_G \sharp* P$
and $A_P \sharp* A_F$
and $A_P \sharp* A_G$
and $A_P \sharp* \Psi_F$
and $A_P \sharp* \Psi_G$

shows $\Psi_G \rhd P \longmapsto \tau \prec P'$
⟨*proof*⟩

lemma *transferFrame*:
fixes Ψ_F :: *'b*
and P :: *('a, 'b, 'c) psi*
and α :: *'a action*
and P' :: *('a, 'b, 'c) psi*
and A_F :: *name list*
and A_G :: *name list*
and Ψ_G :: *'b*

assumes $\Psi_F \rhd P \longmapsto \alpha \prec P'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $\langle A_F, \Psi_F \otimes \Psi_P \rangle \hookrightarrow_F \langle A_G, \Psi_G \otimes \Psi_P \rangle$
and $A_F \sharp* P$
and $A_G \sharp* P$
and $A_F \sharp* subject\ \alpha$
and $A_G \sharp* subject\ \alpha$
and $A_P \sharp* A_F$
and $A_P \sharp* A_G$
and $A_P \sharp* \Psi_F$
and $A_P \sharp* \Psi_G$

shows $\Psi_G \rhd P \longmapsto \alpha \prec P'$
⟨*proof*⟩

lemma *parCasesInputFrame*[*consumes 7*, *case-names cPar1 cPar2*]:

fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and T :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto M(\!|N|\!) \prec T$
and *extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$
and *distinct* A_{PQ}
and $A_{PQ} \sharp* \Psi$
and $A_{PQ} \sharp* P$
and $A_{PQ} \sharp* Q$
and $A_{PQ} \sharp* M$
and *rPar1*: $\bigwedge P'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$;
distinct A_P; *distinct* A_Q; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$;
$A_P \sharp* \Psi_Q$; $A_Q \sharp* \Psi_P$; $A_P \sharp* A_Q$; $A_{PQ} = A_P @ A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$ *Prop* $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|N|\!) \prec Q'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$;
distinct A_P; *distinct* A_Q; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$;
$A_P \sharp* \Psi_Q$; $A_Q \sharp* \Psi_P$; $A_P \sharp* A_Q$; $A_{PQ} = A_P @ A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$ *Prop* $(P \parallel Q')$
shows *Prop* T
⟨*proof*⟩

lemma *parCasesOutputFrame*[*consumes 11*, *case-names cPar1 cPar2*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and T :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec T$
and *xvec* $\sharp* \Psi$
and *xvec* $\sharp* P$
and *xvec* $\sharp* Q$
and *xvec* $\sharp* M$
and *extractFrame*$(P \parallel Q) = \langle A_{PQ}, \Psi_{PQ}\rangle$
and *distinct* A_{PQ}
and $A_{PQ} \sharp* \Psi$

and $A_{PQ} \sharp* P$
and $A_{PQ} \sharp* Q$
and $A_{PQ} \sharp* M$
and *rPar1*: $\bigwedge P'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P';$ *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_P; *distinct* A_Q; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$; $A_P \sharp* \Psi_Q$; $A_Q \sharp* \Psi_P$; $A_P \sharp* A_Q$; $A_{PQ} = A_P @ A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$ *Prop* $(P' \parallel Q)$
and *rPar2*: $\bigwedge Q'\ A_P\ \Psi_P\ A_Q\ \Psi_Q.\ [\![\Psi \otimes \Psi_P \rhd Q \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec Q';$ *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_P; *distinct* A_Q; $A_P \sharp* \Psi$; $A_P \sharp* P$; $A_P \sharp* Q$; $A_P \sharp* M$; $A_Q \sharp* \Psi$; $A_Q \sharp* P$; $A_Q \sharp* Q$; $A_Q \sharp* M$; $A_P \sharp* \Psi_Q$; $A_Q \sharp* \Psi_P$; $A_P \sharp* A_Q$; $A_{PQ} = A_P @ A_Q$; $\Psi_{PQ} = \Psi_P \otimes \Psi_Q]\!] \Longrightarrow$ *Prop* $(P \parallel Q')$
shows *Prop* T
⟨*proof*⟩

inductive *bangPred* :: $('a, 'b, 'c)$ *psi* ⇒ $('a, 'b, 'c)$ *psi* ⇒ *bool*
where
aux1: *bangPred* P $(!P)$
| *aux2*: *bangPred* P $(P \parallel !P)$

lemma *bangInduct*[*consumes 1*, *case-names cPar1 cPar2 cComm1 cComm2 cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rs* :: $('a, 'b, 'c)$ *residual*
and *Prop* :: $'d$::*fs-name* ⇒ $'b$ ⇒ $('a, 'b, 'c)$ *psi* ⇒ $('a, 'b, 'c)$ *residual* ⇒ *bool*
and C :: $'d$

assumes $\Psi \rhd !P \longmapsto Rs$
and *rPar1*: $\bigwedge \alpha\ P'\ C.\ [\![\Psi \rhd P \longmapsto \alpha \prec P';$ *bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp*$ *subject* α; *bn* $\alpha \sharp* C$; *distinct*(*bn* α)$]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (P \parallel !P)\ (\alpha \prec (P' \parallel !P))$
and *rPar2*: $\bigwedge \alpha\ P'\ C.\ [\![\Psi \rhd !P \longmapsto \alpha \prec P';$ *bn* $\alpha \sharp* \Psi$; *bn* $\alpha \sharp* P$; *bn* $\alpha \sharp*$ *subject* α; *bn* $\alpha \sharp* C$; *distinct*(*bn* α)$;$ $\bigwedge C.$ *Prop* $C\ \Psi\ (!P)\ (\alpha \prec P')]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (P \parallel !P)\ (\alpha \prec (P \parallel P'))$
and *rComm1*: $\bigwedge M\ N\ P'\ K\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \Psi \rhd !P \longmapsto K(\!|\nu* xvec|\!)\langle N\rangle \prec P'';\ \bigwedge C.$ *Prop* $C\ \Psi\ (!P)\ (K(\!|\nu* xvec|\!)\langle N\rangle \prec P'');\ \Psi \vdash M \leftrightarrow K;$ *xvec* $\sharp* \Psi$; *xvec* $\sharp* P$; *xvec* $\sharp* M$; *xvec* $\sharp* K$; *xvec* $\sharp* C$; *distinct xvec*$]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (P \parallel !P)\ (\tau \prec (\!|\nu* xvec|\!)(P' \parallel P''))$
and *rComm2*: $\bigwedge M\ xvec\ N\ P'\ K\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu* xvec|\!)\langle N\rangle \prec P';\ \Psi \rhd !P \longmapsto K(\!|N|\!) \prec P'';\ \bigwedge C.$ *Prop* $C\ \Psi\ (!P)\ (K(\!|N|\!) \prec P'');\ \Psi \vdash M \leftrightarrow K;$ *xvec* $\sharp* \Psi$; *xvec* $\sharp* P$; *xvec* $\sharp* M$; *xvec* $\sharp* K$; *xvec* $\sharp* C$; *distinct xvec*$]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (P \parallel !P)\ (\tau \prec (\!|\nu* xvec|\!)(P' \parallel P''))$
and *rBang*: $\bigwedge Rs\ C.\ [\![\Psi \rhd P \parallel !P \longmapsto Rs;\ \bigwedge C.$ *Prop* $C\ \Psi\ (P \parallel !P)\ Rs;$ *guarded* $P]\!] \Longrightarrow$ *Prop* $C\ \Psi\ (!P)\ Rs$
shows *Prop* $C\ \Psi\ (!P)\ Rs$

⟨*proof*⟩

lemma *bangInputInduct*[*consumes 1*, *case-names cPar1 cPar2 cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow 'a \Rightarrow 'a \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*

assumes $\Psi \rhd !P \longmapsto M(\!|N|\!) \prec P'$
and *rPar1*: $\bigwedge P'.\ \Psi \rhd P \longmapsto M(\!|N|\!) \prec P' \Longrightarrow$ *Prop* Ψ $(P \parallel !P)$ M N $(P' \parallel !P)$
and *rPar2*: $\bigwedge P'.\ [\![\Psi \rhd !P \longmapsto M(\!|N|\!) \prec P';$ *Prop* Ψ $(!P)$ M N $P']\!] \Longrightarrow$ *Prop* Ψ $(P \parallel !P)$ M N $(P \parallel P')$
and *rBang*: $\bigwedge P'.\ [\![\Psi \rhd P \parallel !P \longmapsto M(\!|N|\!) \prec P';$ *Prop* Ψ $(P \parallel !P)$ M N $P';$ *guarded* $P]\!] \Longrightarrow$ *Prop* Ψ $(!P)$ M N P'
shows *Prop* Ψ $(!P)$ M N P'
⟨*proof*⟩

lemma *bangOutputInduct*[*consumes 1*, *case-names cPar1 cPar2 cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'d$::*fs-name* $\Rightarrow 'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow 'a \Rightarrow ('a, 'b, 'c)$ *boundOutput* $\Rightarrow$ *bool*
and C :: $'d$

assumes $\Psi \rhd !P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and *rPar1*: $\bigwedge xvec\ N\ P'\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';$ *xvec* $\sharp*$ $\Psi;$ *xvec* $\sharp*$ $P;$ *xvec* $\sharp*$ $M;$ *xvec* $\sharp*$ $C;$ *distinct xvec*$]\!] \Longrightarrow$ *Prop* C Ψ $(P \parallel !P)$ M $((\!|\nu * xvec|\!)N \prec' (P' \parallel !P))$
and *rPar2*: $\bigwedge xvec\ N\ P'\ C.\ [\![\Psi \rhd !P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ \bigwedge C.$ *Prop* C Ψ $(!P)$ M $((\!|\nu * xvec|\!)N \prec' P');$ *xvec* $\sharp*$ $\Psi;$ *xvec* $\sharp*$ $P;$ *xvec* $\sharp*$ $M;$ *xvec* $\sharp*$ $C;$ *distinct xvec*$]\!] \Longrightarrow$
Prop C Ψ $(P \parallel !P)$ M $((\!|\nu * xvec|\!)N \prec' (P \parallel P'))$
and *rBang*: $\bigwedge B\ C.\ [\![\Psi \rhd P \parallel !P \longmapsto (ROut\ M\ B);\ \bigwedge C.$ *Prop* C Ψ $(P \parallel !P)$ M $B;$ *guarded* $P]\!] \Longrightarrow$ *Prop* C Ψ $(!P)$ M B

shows *Prop* C Ψ $(!P)$ M $((\!|\nu * xvec|\!)N \prec' P')$
⟨*proof*⟩

lemma *bangTauInduct*[*consumes 1*, *case-names cPar1 cPar2 cComm1 cComm2 cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*

and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'d$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
and C :: $'d$

assumes $\Psi \rhd\, !P \longmapsto \tau \prec P'$
and *rPar1*: $\bigwedge P'\ C.\ \Psi \rhd P \longmapsto \tau \prec P' \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ (P' \parallel\, !P)$
and *rPar2*: $\bigwedge P'\ C.\ [\![\Psi \rhd\, !P \longmapsto \tau \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ P']\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ (P \parallel P')$
and *rComm1*: $\bigwedge M\ N\ P'\ K\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \Psi \rhd\, !P \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'';\ \Psi \vdash M \leftrightarrow K;\ xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ ((\!|\nu{*}xvec|\!)(P' \parallel P''))$
and *rComm2*: $\bigwedge M\ N\ P'\ K\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P';\ \Psi \rhd\, !P \longmapsto K(\!|N|\!) \prec P'';\ \Psi \vdash M \leftrightarrow K;\ xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ ((\!|\nu{*}xvec|\!)(P' \parallel P''))$
and *rBang*: $\bigwedge P'\ C.\ [\![\Psi \rhd P \parallel\, !P \longmapsto \tau \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel\, !P)\ P';\ guarded\ P]\!] \Longrightarrow Prop\ C\ \Psi\ (!P)\ P'$

shows $Prop\ C\ \Psi\ (!P)\ P'$
⟨*proof*⟩

lemma *bangInduct'*[*consumes 2*, *case-names cAlpha cPar1 cPar2 cComm1 cComm2 cBang*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and *Prop* :: $'d$::*fs-name* $\Rightarrow$ $'b \Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ $'a$ *action* $\Rightarrow ('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
and C :: $'d$::*fs-name*

assumes $\Psi \rhd\, !P \longmapsto \alpha \prec P'$
and $bn\ \alpha \sharp* subject\ \alpha$
and *rAlpha*: $\bigwedge \alpha\ P'\ p\ C.\ [\![bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ set\ p \subseteq set(bn\ \alpha) \times set(bn(p \cdot \alpha));\ distinctPerm\ p;\ bn(p \cdot \alpha) \sharp* \alpha;\ bn(p \cdot \alpha) \sharp* P';\ Prop\ C\ \Psi\ (P \parallel\, !P)\ \alpha\ P']\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ (p \cdot \alpha)\ (p \cdot P')$
and *rPar1*: $\bigwedge \alpha\ P'\ C.\ [\![\Psi \rhd P \longmapsto \alpha \prec P';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ distinct(bn\ \alpha)]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ \alpha\ (P' \parallel\, !P)$
and *rPar2*: $\bigwedge \alpha\ P'\ C.\ [\![\Psi \rhd\, !P \longmapsto \alpha \prec P';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ \alpha\ P';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ distinct(bn\ \alpha)]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel\, !P)\ \alpha\ (P \parallel P')$

and *rComm1*: $\bigwedge M\ N\ P'\ K\ xvec\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P';\ \Psi \rhd\ !P \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec P'';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ (K(\!|\nu{*}xvec|\!)\langle N\rangle)\ P'';\ \Psi \vdash M \leftrightarrow K;$

$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel !P)\ (\tau)\ ((\!|\nu{*}xvec|\!)(P' \parallel P''))$

and *rComm2*: $\bigwedge M\ xvec\ N\ P'\ K\ P''\ C.\ [\![\Psi \rhd P \longmapsto M(\!|\nu{*}xvec|\!)\langle N\rangle \prec P';\ \Psi \rhd\ !P \longmapsto K(\!|N|\!) \prec P'';\ \bigwedge C.\ Prop\ C\ \Psi\ (!P)\ (K(\!|N|\!))\ P'';\ \Psi \vdash M \leftrightarrow K;$

$xvec \sharp* \Psi;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* C;\ distinct\ xvec]\!] \Longrightarrow Prop\ C\ \Psi\ (P \parallel !P)\ (\tau)\ ((\!|\nu{*}xvec|\!)(P' \parallel P''))$

and *rBang*: $\bigwedge \alpha\ P'\ C.$

$[\![\Psi \rhd P \parallel !P \longmapsto \alpha \prec P';\ guarded\ P;\ \bigwedge C.\ Prop\ C\ \Psi\ (P \parallel !P)\ \alpha\ P';\ guarded\ P;\ distinct(bn\ \alpha)]\!] \Longrightarrow$

$Prop\ C\ \Psi\ (!P)\ \alpha\ P'$

shows $Prop\ C\ \Psi\ (!P)\ \alpha\ P'$

⟨*proof*⟩

lemma *comm1Aux*:

fixes Ψ :: $'b$

and Ψ_Q :: $'b$

and R :: $('a, 'b, 'c)\ psi$

and K :: $'a$

and $xvec$ :: *name list*

and N :: $'a$

and R' :: $('a, 'b, 'c)\ psi$

and A_R :: *name list*

and Ψ_R :: $'b$

and P :: $('a, 'b, 'c)\ psi$

and M :: $'a$

and L :: $'a$

and P' :: $('a, 'b, 'c)\ psi$

and A_P :: *name list*

and Ψ_P :: $'b$

and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto K(\!|\nu{*}xvec|\!)\langle N\rangle \prec R'$

and *FrR*: $extractFrame\ R = \langle A_R, \Psi_R\rangle$

and *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|L|\!) \prec P'$

and *MeqK*: $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$

and *PeqQ*: $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$

and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P\rangle$

and *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$

and $distinct\ A_P$

and $distinct\ A_R$

and $A_R \sharp* A_P$

and $A_R \sharp* A_Q$

and $A_R \sharp* \Psi$

and $A_R \sharp* P$

and $A_R \sharp* Q$

and $A_R \sharp* R$

and $A_R \sharp* K$
and $A_P \sharp* \Psi$
and $A_P \sharp* R$
and $A_P \sharp* P$
and $A_P \sharp* M$
and $A_Q \sharp* R$
and $A_Q \sharp* M$

obtains K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto K'(\!|\nu*xvec|\!)\langle N\rangle \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $A_R \sharp* K'$
⟨*proof*⟩

lemma *comm2Aux*:
fixes Ψ :: $'b$
and Ψ_Q :: $'b$
and R :: $('a, 'b, 'c)$ *psi*
and K :: $'a$
and N :: $'a$
and R' :: $('a, 'b, 'c)$ *psi*
and A_R :: *name list*
and Ψ_R :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and P' :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*
and Ψ_P :: $'b$
and A_Q :: *name list*

assumes *RTrans*: $\Psi \otimes \Psi_Q \rhd R \longmapsto K(\!|N|\!) \prec R'$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and *PTrans*: $\Psi \otimes \Psi_R \rhd P \longmapsto M(\!|\nu*xvec|\!)\langle N\rangle \prec P'$
and *MeqK*: $\Psi \otimes \Psi_Q \otimes \Psi_R \vdash M \leftrightarrow K$
and *QimpP*: $\langle A_Q, (\Psi \otimes \Psi_Q) \otimes \Psi_R\rangle \hookrightarrow_F \langle A_P, (\Psi \otimes \Psi_P) \otimes \Psi_R\rangle$
and *FrP*: *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and *distinct* A_P
and *distinct* A_R
and $A_R \sharp* A_P$
and $A_R \sharp* A_Q$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$
and $A_R \sharp* R$
and $A_R \sharp* K$
and $A_P \sharp* \Psi$
and $A_P \sharp* R$
and $A_P \sharp* P$
and $A_P \sharp* M$

and $A_Q \sharp* R$
and $A_Q \sharp* M$
and $A_R \sharp* xvec$
and $xvec \sharp* M$

obtains K' **where** $\Psi \otimes \Psi_P \rhd R \longmapsto K'\langle\!|N|\!\rangle \prec R'$ **and** $\Psi \otimes \Psi_P \otimes \Psi_R \vdash M \leftrightarrow K'$ **and** $A_R \sharp* K'$
⟨*proof*⟩

end

end

theory *Simulation*
imports *Semantics*
begin

context *env* **begin**

definition
simulation :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow$
$('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set \Rightarrow$
$('a, 'b, 'c)\ psi \Rightarrow bool$ (‹- ▷ - ⇝[-] -› [80, 80, 80, 80] 80)
where
$\Psi \rhd P \rightsquigarrow[Rel]\ Q \equiv \forall \alpha\ Q'.\ \Psi \rhd Q \longmapsto \alpha \prec Q' \longrightarrow bn\ \alpha \sharp* \Psi \longrightarrow bn\ \alpha \sharp* P \longrightarrow (\exists P'.\ \Psi \rhd P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel)$

abbreviation
simulationNilJudge (‹- ⇝[-] -› [80, 80, 80] 80) **where** $P \rightsquigarrow[Rel]\ Q \equiv SBottom' \rhd P \rightsquigarrow[Rel]\ Q$

lemma *simI*[*consumes 1*, *case-names cSim*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and *Sim*: $\bigwedge \alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto \alpha \prec Q'; bn\ \alpha \sharp* P; bn\ \alpha \sharp* Q; bn\ \alpha \sharp* \Psi; distinct(bn\ \alpha);$
$bn\ \alpha \sharp* (subject\ \alpha); bn\ \alpha \sharp* C]\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

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

lemma *simI2*[*case-names cSim*]:
fixes Ψ :: $'b$

and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Sim*: $\bigwedge\alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto\alpha \prec Q';\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* \Psi;\ distinct(bn\ \alpha)]\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto\alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

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

lemma *simIChainFresh*[*consumes 4*, *case-names cSim*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and *xvec* $\sharp*$ Ψ
and *xvec* $\sharp*$ P
and *xvec* $\sharp*$ Q
and *Sim*: $\bigwedge\alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto\alpha \prec Q';\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* subject\ \alpha;\ distinct(bn\ \alpha);\ bn\ \alpha \sharp* C;\ xvec \sharp* \alpha;\ xvec \sharp* Q']\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto\alpha \prec P' \wedge (\Psi, P', Q') \in Rel$
shows $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
⟨*proof*⟩

lemma *simIFresh*[*consumes 4*, *case-names cSim*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and x :: *name*
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp P$
and $x \sharp Q$
and $\bigwedge\alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto\alpha \prec Q';\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* subject\ \alpha;\ distinct(bn\ \alpha);\ bn\ \alpha \sharp* C;\ x \sharp \alpha;\ x \sharp Q']\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto\alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

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

lemma *simE*:
fixes F :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \rightsquigarrow[Rel]\ Q$

shows $\bigwedge \alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto \alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P]\!] \Longrightarrow \exists P'.\ \Psi \rhd P \longmapsto \alpha \prec P' \wedge (\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *simClosedAux*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and p :: $name\ prm$

assumes *EqvtRel*: $eqvt\ Rel$
and *PSimQ*: $\Psi \rhd P \rightsquigarrow[Rel]\ Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow[Rel]\ (p \cdot Q)$
⟨*proof*⟩

lemma *simClosed*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and p :: $name\ prm$

assumes *EqvtRel*: $eqvt\ Rel$

shows $\Psi \rhd P \rightsquigarrow[Rel]\ Q \Longrightarrow (p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow[Rel]\ (p \cdot Q)$
and $P \rightsquigarrow[Rel]\ Q \Longrightarrow (p \cdot P) \rightsquigarrow[Rel]\ (p \cdot Q)$
⟨*proof*⟩

lemma *reflexive*:
fixes Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

assumes $\{(\Psi, P, P) \mid \Psi\ P.\ True\} \subseteq Rel$

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

lemma *transitive*:

fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and R :: $('a, 'b, 'c)\ psi$
and Rel'' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
and *QSimR*: $\Psi \rhd Q \rightsquigarrow[Rel']\ R$
and *Eqvt*: *eqvt* Rel''
and *Set*: $\{(\Psi, P, R) \mid \Psi\ P\ R.\ \exists Q.\ (\Psi, P, Q) \in Rel \wedge (\Psi, Q, R) \in Rel'\} \subseteq Rel''$

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

lemma *statEqSim*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and Ψ' :: $'b$

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
and *eqvt* Rel'
and $\Psi \simeq \Psi'$
and *C1*: $\bigwedge \Psi''\ R\ S\ \Psi'''.\ [\![(\Psi'', R, S) \in Rel;\ \Psi'' \simeq \Psi''']\!] \Longrightarrow (\Psi''', R, S) \in Rel'$

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

lemma *monotonic*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and A :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and B :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

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

shows $\Psi \rhd P \rightsquigarrow[B]\ Q$
⟨*proof*⟩

end

end

theory *Tau-Chain*
  imports *Semantics*
begin

context *env* **begin**

abbreviation *tauChain* :: *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ▷ - ⟹^τ -› [*80*, *80*, *80*] *80*)
where Ψ ▷ *P* ⟹^τ *P'* ≡ (*P*, *P'*) ∈ {(*P*, *P'*). Ψ ▷ *P* ⟼τ ≺ *P'*}⁀*

abbreviation *tauStepChain* :: *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ▷ - ⟹τ -› [*80*, *80*, *80*] *80*)
where Ψ ▷ *P* ⟹τ *P'* ≡ (*P*, *P'*) ∈ {(*P*, *P'*). Ψ ▷ *P* ⟼τ ≺ *P'*}⁀+

abbreviation *tauContextChain* :: (*'a*, *'b*, *'c*) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ⟹^τ -› [*80*, *80*] *80*)
where *P* ⟹^τ *P'* ≡ **1** ▷ *P* ⟹^τ *P'*
abbreviation *tauContextStepChain* :: (*'a*, *'b*, *'c*) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ⟹τ -› [*80*, *80*] *80*)
where *P* ⟹τ *P'* ≡ **1** ▷ *P* ⟹τ *P'*

lemmas *tauChainInduct*[*consumes 1*, *case-names TauBase TauStep*] = *rtrancl.induct*[*of* - - {(*P*, *P'*). Ψ ▷ *P* ⟼τ ≺ *P'*}, *simplified*] **for** Ψ
lemmas *tauStepChainInduct*[*consumes 1*, *case-names TauBase TauStep*] = *trancl.induct*[*of* - - {(*P*, *P'*). Ψ ▷ *P* ⟼τ ≺ *P'*}, *simplified*] **for** Ψ

lemma *tauActTauStepChain*:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *P'* :: (*'a*, *'b*, *'c*) *psi*

  assumes Ψ ▷ *P* ⟼τ ≺ *P'*

  shows Ψ ▷ *P* ⟹τ *P'*
⟨*proof*⟩

lemma *tauActTauChain*:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *P'* :: (*'a*, *'b*, *'c*) *psi*

  assumes Ψ ▷ *P* ⟼τ ≺ *P'*

  shows Ψ ▷ *P* ⟹^τ *P'*
⟨*proof*⟩

lemma *tauStepChainEqvt*[*eqvt*]:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes $\Psi \rhd P \Longrightarrow_\tau P'$

shows $(p \cdot \Psi) \rhd (p \cdot P) \Longrightarrow_\tau (p \cdot P')$
⟨*proof*⟩

lemma *tauChainEqvt*[*eqvt*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau P'$

shows $(p \cdot \Psi) \rhd (p \cdot P) \Longrightarrow\hat{}_\tau (p \cdot P')$
⟨*proof*⟩

lemma *tauStepChainEqvt'*[*eqvt*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

shows $(p \cdot (\Psi \rhd P \Longrightarrow_\tau P')) = (p \cdot \Psi) \rhd (p \cdot P) \Longrightarrow_\tau (p \cdot P')$
⟨*proof*⟩

lemma *tauChainEqvt'*[*eqvt*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

shows $(p \cdot (\Psi \rhd P \Longrightarrow\hat{}_\tau P')) = (p \cdot \Psi) \rhd (p \cdot P) \Longrightarrow\hat{}_\tau (p \cdot P')$
⟨*proof*⟩

lemma *tauStepChainFresh*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \Longrightarrow_\tau P'$
and $x \sharp P$

shows $x \sharp P'$

⟨*proof*⟩

lemma *tauChainFresh*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*

assumes $\Psi \rhd P \Longrightarrow\hat{}_{\tau}\ P'$
and $x \sharp P$

shows $x \sharp P'$
⟨*proof*⟩

lemma *tauStepChainFreshChain*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes $\Psi \rhd P \Longrightarrow_{\tau} P'$
and *xvec* $\sharp*$ P

shows *xvec* $\sharp*$ P'
⟨*proof*⟩

lemma *tauChainFreshChain*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes $\Psi \rhd P \Longrightarrow\hat{}_{\tau}\ P'$
and *xvec* $\sharp*$ P

shows *xvec* $\sharp*$ P'
⟨*proof*⟩

lemma *tauStepChainCase*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and φ :: $'c$
and *Cs* :: $('c \times ('a, 'b, 'c)$ *psi*$)$ *list*

assumes $\Psi \rhd P \Longrightarrow_{\tau} P'$
and (φ, P) *mem Cs*
and $\Psi \vdash \varphi$
and *guarded P*

shows $\Psi \rhd (Cases\ Cs) \Longrightarrow_\tau P'$
⟨*proof*⟩

lemma *tauStepChainResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \Longrightarrow_\tau P'$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!) P \Longrightarrow_\tau (\!|\nu x|\!) P'$
⟨*proof*⟩

lemma *tauChainResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \Longrightarrow\!\hat{}_\tau P'$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!) P \Longrightarrow\!\hat{}_\tau (\!|\nu x|\!) P'$
⟨*proof*⟩

lemma *tauStepChainResChainPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \Longrightarrow_\tau P'$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu * xvec|\!) P \Longrightarrow_\tau (\!|\nu * xvec|\!) P'$
⟨*proof*⟩

lemma *tauChainResChainPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \Longrightarrow\!\hat{}_\tau P'$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu{*}xvec|\!)P \Longrightarrow\hat{}_\tau\ (\!|\nu{*}xvec|\!)P'$
⟨*proof*⟩

lemma *tauStepChainPar1*:
fixes Ψ :: $'b$
and Ψ_Q :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*

assumes $\Psi \otimes \Psi_Q \rhd P \Longrightarrow_\tau P'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$

shows $\Psi \rhd P \parallel Q \Longrightarrow_\tau P' \parallel Q$
⟨*proof*⟩

lemma *tauChainPar1*:
fixes Ψ :: $'b$
and Ψ_Q :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*

assumes $\Psi \otimes \Psi_Q \rhd P \Longrightarrow\hat{}_\tau P'$
and *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$

shows $\Psi \rhd P \parallel Q \Longrightarrow\hat{}_\tau P' \parallel Q$
⟨*proof*⟩

lemma *tauStepChainPar2*:
fixes Ψ :: $'b$
and Ψ_P :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and Q' :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*

assumes $\Psi \otimes \Psi_P \rhd Q \Longrightarrow_\tau Q'$
and *extractFrame* $P = \langle A_P, \Psi_P\rangle$
and $A_P \sharp* \Psi$
and $A_P \sharp* Q$

shows $\Psi \rhd P \parallel Q \Longrightarrow_\tau P \parallel Q'$

⟨*proof*⟩

lemma *tauChainPar2*:
fixes Ψ :: $'b$
and Ψ_P :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and Q' :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and A_P :: *name list*

assumes $\Psi \otimes \Psi_P \rhd Q \Longrightarrow\hat{}_\tau\ Q'$
and *extractFrame* $P = \langle A_P, \Psi_P \rangle$
and $A_P \sharp* \Psi$
and $A_P \sharp* Q$

shows $\Psi \rhd P \parallel Q \Longrightarrow\hat{}_\tau\ P \parallel Q'$
⟨*proof*⟩

lemma *tauStepChainBang*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \parallel !P \Longrightarrow_\tau P'$
and *guarded* P

shows $\Psi \rhd !P \Longrightarrow_\tau P'$
⟨*proof*⟩

lemma *tauStepChainStatEq*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and Ψ' :: $'b$

assumes $\Psi \rhd P \Longrightarrow_\tau P'$
and $\Psi \simeq \Psi'$

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

lemma *tauChainStatEq*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and Ψ' :: $'b$

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P'$
and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \Longrightarrow\hat{}_\tau\ P'$
⟨*proof*⟩

definition *weakTransition* :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow 'a\ action \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow bool$ $(\langle\text{-} : \text{-} \rhd \text{-} \Longrightarrow \text{-} \prec \text{-}\rangle\ [80, 80, 80, 80, 80]\ 80)$
where
$\Psi : Q \rhd P \Longrightarrow \alpha \prec P' \equiv \exists P''.\ \Psi \rhd P \Longrightarrow\hat{}_\tau\ P'' \wedge (insertAssertion\ (extractFrame\ Q)\ \Psi) \hookrightarrow_F (insertAssertion\ (extractFrame\ P'')\ \Psi) \wedge \Psi \rhd P'' \longmapsto \alpha \prec P'$

lemma *weakTransitionI*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)\ psi$
and P :: $('a, 'b, 'c)\ psi$
and P'' :: $('a, 'b, 'c)\ psi$
and α :: $'a\ action$
and P' :: $('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P''$
and $insertAssertion\ (extractFrame\ Q)\ \Psi \hookrightarrow_F insertAssertion\ (extractFrame\ P'')\ \Psi$
and $\Psi \rhd P'' \longmapsto \alpha \prec P'$

shows $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *weakTransitionE*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)\ psi$
and P :: $('a, 'b, 'c)\ psi$
and α :: $'a\ action$
and P' :: $('a, 'b, 'c)\ psi$

assumes $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$

obtains P'' **where** $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P''$ **and** $insertAssertion\ (extractFrame\ Q)\ \Psi \hookrightarrow_F insertAssertion\ (extractFrame\ P'')\ \Psi$
and $\Psi \rhd P'' \longmapsto \alpha \prec P'$
⟨*proof*⟩

lemma *weakTransitionClosed*[*eqvt*]:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)\ psi$
and P :: $('a, 'b, 'c)\ psi$
and α :: $'a\ action$
and P' :: $('a, 'b, 'c)\ psi$
and p :: $name\ prm$

assumes $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$

shows $(p \cdot \Psi) : (p \cdot Q) \rhd (p \cdot P) \Longrightarrow (p \cdot \alpha) \prec (p \cdot P')$
⟨*proof*⟩

lemma *weakOutputAlpha*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*
and *yvec* :: *name list*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow M\langle\!|\nu * (p \cdot xvec)|\!\rangle\langle(p \cdot N)\rangle \prec P'$
and S: *set* $p \subseteq$ *set xvec* $\times$ *set*$(p \cdot xvec)$
and *distinctPerm p*
and *xvec* $\sharp*$ P
and *xvec* $\sharp*$ $(p \cdot xvec)$
and $(p \cdot xvec)$ $\sharp*$ M
and *distinct xvec*

shows $\Psi : Q \rhd P \Longrightarrow M\langle\!|\nu * xvec|\!\rangle\langle N\rangle \prec (p \cdot P')$
⟨*proof*⟩

lemma *weakFreshDerivative*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and $x \sharp P$
and $x \sharp \alpha$
and *bn* α $\sharp*$ *subject* α
and *distinct*$(bn\ \alpha)$

shows $x \sharp P'$
⟨*proof*⟩

lemma *weakFreshChainDerivative*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*

and P' :: $('a, 'b, 'c)$ *psi*
and *yvec* :: *name list*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and *yvec* $\sharp*$ P
and *yvec* $\sharp*$ α
and *bn* α $\sharp*$ *subject* α
and *distinct*(*bn* α)

shows *yvec* $\sharp*$ P'
⟨*proof*⟩

lemma *weakInputFreshDerivative*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and x :: *name*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow M(\!|N|\!) \prec P'$
and $x \sharp P$
and $x \sharp N$

shows $x \sharp P'$
⟨*proof*⟩

lemma *weakInputFreshChainDerivative*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow M(\!|N|\!) \prec P'$
and *xvec* $\sharp*$ P
and *xvec* $\sharp*$ N

shows *xvec* $\sharp*$ P'
⟨*proof*⟩

lemma *weakOutputFreshDerivative*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$

and $xvec :: name\ list$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $PTrans$: $\Psi : Q \rhd P \Longrightarrow M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and $x \sharp P$
and $x \sharp xvec$
and $xvec \sharp* M$
and $distinct\ xvec$

shows $x \sharp N$
and $x \sharp P'$
⟨*proof*⟩

lemma *weakOutputFreshChainDerivative*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $yvec :: name\ list$

assumes $PTrans$: $\Psi : Q \rhd P \Longrightarrow M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and $yvec \sharp* P$
and $xvec \sharp* yvec$
and $xvec \sharp* M$
and $distinct\ xvec$

shows $yvec \sharp* N$
and $yvec \sharp* P'$
⟨*proof*⟩

lemma *weakOutputPermSubject*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $yvec :: name\ list$
and $zvec :: name\ list$

assumes $PTrans$: $\Psi : Q \rhd P \Longrightarrow M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and S: $set\ p \subseteq set\ yvec \times set\ zvec$
and $yvec \sharp* \Psi$

and $zvec \sharp* \Psi$
and $yvec \sharp* P$
and $zvec \sharp* P$

shows $\Psi : Q \rhd P \Longrightarrow (p \cdot M)(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *weakInputPermSubject*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$
and $yvec :: name\ list$
and $zvec :: name\ list$

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow M(\!|N|\!) \prec P'$
and S: $set\ p \subseteq set\ yvec \times set\ zvec$
and $yvec \sharp* \Psi$
and $zvec \sharp* \Psi$
and $yvec \sharp* P$
and $zvec \sharp* P$

shows $\Psi : Q \rhd P \Longrightarrow (p \cdot M)(\!|N|\!) \prec P'$
⟨*proof*⟩

lemma *weakInput*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $K :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $Tvec :: 'a\ list$
and $P :: ('a, 'b, 'c)\ psi$

assumes $\Psi \vdash M \leftrightarrow K$
and $distinct\ xvec$
and $set\ xvec \subseteq supp\ N$
and $length\ xvec = length\ Tvec$
and $Qeq\Psi$: $insertAssertion\ (extractFrame\ Q)\ \Psi \hookrightarrow_F \langle \varepsilon, \Psi \otimes \mathbf{1}\rangle$

shows $\Psi : Q \rhd M(\!|\lambda * xvec\ N|\!).P \Longrightarrow K(\!|(N[xvec ::= Tvec])|\!) \prec P[xvec ::= Tvec]$
⟨*proof*⟩

lemma *weakOutput*:
fixes $\Psi :: 'b$

and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and K :: $'a$
and N :: $'a$
and P :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \vdash M \leftrightarrow K$
and *Qeq*Ψ: *insertAssertion* (*extractFrame* Q) $\Psi \hookrightarrow_F \langle \varepsilon, \Psi \otimes \mathbf{1} \rangle$

shows $\Psi : Q \rhd M\langle N \rangle.P \Longrightarrow K\langle N \rangle \prec P$
⟨*proof*⟩

lemma *insertGuardedAssertion*:
fixes P :: $('a, 'b, 'c)$ *psi*

assumes *guarded* P

shows *insertAssertion*(*extractFrame* P) $\Psi \simeq_F \langle \varepsilon, \Psi \otimes \mathbf{1} \rangle$
⟨*proof*⟩

lemma *weakCase*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and (φ, P) *mem CsP*
and $\Psi \vdash \varphi$
and *guarded* P
and *RImpQ*: *insertAssertion* (*extractFrame* R) $\Psi \hookrightarrow_F$ *insertAssertion* (*extractFrame* Q) Ψ
and *ImpR*: *insertAssertion* (*extractFrame* R) $\Psi \hookrightarrow_F \langle \varepsilon, \Psi \rangle$

shows $\Psi : R \rhd Cases\ CsP \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *weakOpen*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and *yvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow M(\!|\nu*(xvec@yvec)|\!)\langle N\rangle \prec P'$
and $x \in supp\ N$
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp yvec$

shows $\Psi : (\!|\nu x|\!)Q \rhd (\!|\nu x|\!)P \Longrightarrow M(\!|\nu*(xvec@x\#yvec)|\!)\langle N\rangle \prec P'$
⟨*proof*⟩

lemma *weakScope*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and $x \sharp \Psi$
and $x \sharp \alpha$

shows $\Psi : (\!|\nu x|\!)Q \rhd (\!|\nu x|\!)P \Longrightarrow \alpha \prec (\!|\nu x|\!)P'$
⟨*proof*⟩

lemma *weakPar1*:
fixes $\Psi :: 'b$
and $R :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $A_Q :: name\ list$
and $\Psi_Q :: 'b$

assumes *PTrans*: $\Psi \otimes \Psi_Q : R \rhd P \Longrightarrow \alpha \prec P'$
and *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q\rangle$
and $bn\ \alpha \sharp* Q$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$
and $A_Q \sharp* \alpha$
and $A_Q \sharp* R$

shows $\Psi : R \parallel Q \rhd P \parallel Q \Longrightarrow \alpha \prec P' \parallel Q$
⟨*proof*⟩

lemma *weakPar2*:
fixes $\Psi :: 'b$
and $R :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $Q' :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $A_P :: name\ list$
and $\Psi_P :: 'b$

assumes *QTrans*: $\Psi \otimes \Psi_P : R \rhd Q \Longrightarrow \alpha \prec Q'$
and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and $bn\ \alpha \sharp* P$
and $A_P \sharp* \Psi$
and $A_P \sharp* Q$
and $A_P \sharp* \alpha$
and $A_P \sharp* R$

shows $\Psi : P \parallel R \rhd P \parallel Q \Longrightarrow \alpha \prec P \parallel Q'$
⟨*proof*⟩

lemma *weakComm1*:
fixes $\Psi :: 'b$
and $R :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $A_Q :: name\ list$
and $\Psi_Q :: 'b$

assumes *PTrans*: $\Psi \otimes \Psi_Q : R \rhd P \Longrightarrow M(\!|N|\!) \prec P'$
and *FrR*: $extractFrame\ R = \langle A_R, \Psi_R \rangle$
and *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N \rangle \prec Q'$
and *FrQ*: $extractFrame\ Q = \langle A_Q, \Psi_Q \rangle$
and *MeqK*: $\Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_R \sharp* A_Q$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$
and $A_Q \sharp* Q$
and $A_Q \sharp* R$
and $A_Q \sharp* K$
and $xvec \sharp* P$

shows $\Psi \rhd P \parallel Q \Longrightarrow_\tau (\!|\nu * xvec|\!)(P' \parallel Q'))$
⟨*proof*⟩

lemma *weakComm2*:
fixes Ψ :: $'b$
and R :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and A_Q :: *name list*
and Ψ_Q :: $'b$

assumes *PTrans*: $\Psi \otimes \Psi_Q : R \rhd P \Longrightarrow M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and *QTrans*: $\Psi \otimes \Psi_R \rhd Q \longmapsto K(\!|N|\!) \prec Q'$
and *FrQ*: *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$
and *MeqK*: $\Psi \otimes \Psi_R \otimes \Psi_Q \vdash M \leftrightarrow K$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$
and $A_R \sharp* R$
and $A_R \sharp* M$
and $A_R \sharp* A_Q$
and $A_Q \sharp* \Psi$
and $A_Q \sharp* P$
and $A_Q \sharp* Q$
and $A_Q \sharp* R$
and $A_Q \sharp* K$
and $xvec \sharp* Q$
and $xvec \sharp* M$
and $xvec \sharp* A_Q$
and $xvec \sharp* A_R$

shows $\Psi \rhd P \parallel Q \Longrightarrow_\tau (\!|\nu * xvec|\!)(P' \parallel Q')$
⟨*proof*⟩

lemma *frameImpIntComposition*:
fixes Ψ :: $'b$
and Ψ' :: $'b$
and A_F :: *name list*
and Ψ_F :: $'b$

assumes $\Psi \simeq \Psi'$

shows $\langle A_F, \Psi \otimes \Psi_F\rangle \hookrightarrow_F \langle A_F, \Psi' \otimes \Psi_F\rangle$
⟨*proof*⟩

lemma *insertAssertionStatImp*:
fixes F :: $'b$ *frame*
and Ψ :: $'b$

and $G :: 'b\ frame$
and $\Psi' :: 'b$

assumes *FeqG*: $insertAssertion\ F\ \Psi \hookrightarrow_F insertAssertion\ G\ \Psi$
and $\Psi \simeq \Psi'$

shows $insertAssertion\ F\ \Psi' \hookrightarrow_F insertAssertion\ G\ \Psi'$
⟨*proof*⟩

lemma *insertAssertionStatEq*:
fixes $F :: 'b\ frame$
and $\Psi :: 'b$
and $G :: 'b\ frame$
and $\Psi' :: 'b$

assumes *FeqG*: $insertAssertion\ F\ \Psi \simeq_F insertAssertion\ G\ \Psi$
and $\Psi \simeq \Psi'$

shows $insertAssertion\ F\ \Psi' \simeq_F insertAssertion\ G\ \Psi'$
⟨*proof*⟩

lemma *weakTransitionStatEq*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and $\Psi \simeq \Psi'$

shows $\Psi' : Q \rhd P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *transitionWeakTransition*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \longmapsto \alpha \prec P'$
and $insertAssertion(extractFrame\ Q)\ \Psi \hookrightarrow_F insertAssertion(extractFrame\ P)\ \Psi$

shows $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *weakPar1Guarded*:
fixes Ψ :: $'b$
and R :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

assumes *PTrans*: $\Psi : R \rhd P \Longrightarrow \alpha \prec P'$
and $bn\ \alpha \sharp* Q$
and *guarded* Q

shows $\Psi : (R \parallel Q) \rhd P \parallel Q \Longrightarrow \alpha \prec P' \parallel Q$
⟨*proof*⟩

lemma *weakBang*:
fixes Ψ :: $'b$
and R :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

assumes *PTrans*: $\Psi : R \rhd P \parallel !P \Longrightarrow \alpha \prec P'$
and *guarded* P

shows $\Psi : R \rhd !P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *weakTransitionFrameImp*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and *insertAssertion*(*extractFrame* R) $\Psi \hookrightarrow_F$ *insertAssertion*(*extractFrame* Q) Ψ

shows $\Psi : R \rhd P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *guardedFrameStatEq*:
fixes P :: $('a, 'b, 'c)$ *psi*

assumes *guarded P*

shows *extractFrame P* $\simeq_F$ $\langle \varepsilon, \mathbf{1} \rangle$
⟨*proof*⟩

lemma *weakGuardedTransition*:
fixes Ψ :: *'b*
and Q :: *('a, 'b, 'c) psi*
and P :: *('a, 'b, 'c) psi*
and α :: *'a action*
and P' :: *('a, 'b, 'c) psi*
and R :: *('a, 'b, 'c) psi*

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and *guarded Q*

shows $\Psi : \mathbf{0} \rhd P \Longrightarrow \alpha \prec P'$
⟨*proof*⟩

lemma *expandTauChainFrame*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and P' :: *('a, 'b, 'c) psi*
and A_P :: *name list*
and Ψ_P :: *'b*
and C :: *'d::fs-name*

assumes *PChain*: $\Psi \rhd P \Longrightarrow\hat{}_\tau P'$
and *FrP*: *extractFrame P* $= \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and $A_P \sharp* P$
and $A_P \sharp* C$

obtains Ψ' A_P' Ψ_P' **where** *extractFrame* $P' = \langle A_P', \Psi_P' \rangle$ **and** $\Psi_P \otimes \Psi' \simeq \Psi_P'$ **and** $A_P' \sharp* P'$ **and** $A_P' \sharp* C$ **and** *distinct* A_P'
⟨*proof*⟩

lemma *frameIntImpComposition*:
fixes Ψ :: *'b*
and Ψ' :: *'b*
and A_F :: *name list*
and Ψ_F :: *'b*

assumes $\Psi \simeq \Psi'$

shows $\langle A_F, \Psi \otimes \Psi_F \rangle \hookrightarrow_F \langle A_F, \Psi' \otimes \Psi_F \rangle$
⟨*proof*⟩

lemma *tauChainInduct2*[*consumes 1*, *case-names TauBase TauStep*]:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *PChain*: $\Psi \rhd P \Longrightarrow^{\hat{}}_{\tau} P'$
and *cBase*: $\bigwedge P.\ Prop\ P\ P$
and *cStep*: $\bigwedge P\ P'\ P''.\ [\![\Psi \rhd P' \longmapsto \tau \prec P''; \Psi \rhd P \Longrightarrow^{\hat{}}_{\tau} P'; Prop\ P\ P']\!] \Longrightarrow Prop\ P\ P''$

shows *Prop P P'*
⟨*proof*⟩

lemma *tauStepChainInduct2*[*consumes 1*, *case-names TauBase TauStep*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *PChain*: $\Psi \rhd P \Longrightarrow_{\tau} P'$
and *cBase*: $\bigwedge P\ P'.\ \Psi \rhd P \longmapsto \tau \prec P' \Longrightarrow Prop\ P\ P'$
and *cStep*: $\bigwedge P\ P'\ P''.\ [\![\Psi \rhd P' \longmapsto \tau \prec P''; \Psi \rhd P \Longrightarrow_{\tau} P'; Prop\ P\ P']\!] \Longrightarrow Prop\ P\ P''$

shows *Prop P P'*
⟨*proof*⟩

lemma *weakTransferTauChainFrame*:
fixes $\Psi_F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$
and $A_P :: name\ list$
and $\Psi_P :: 'b$
and $A_F :: name\ list$
and $A_G :: name\ list$
and $\Psi_G :: 'b$

assumes *PChain*: $\Psi_F \rhd P \Longrightarrow^{\hat{}}_{\tau} P'$
and *FrP*: $extractFrame\ P = \langle A_P, \Psi_P \rangle$
and *distinct* A_P
and *FeqG*: $\bigwedge \Psi.\ insertAssertion\ (\langle A_F, \Psi_F \otimes \Psi_P \rangle)\ \Psi \hookrightarrow_F insertAssertion\ (\langle A_G, \Psi_G \otimes \Psi_P \rangle)\ \Psi$
and $A_F \sharp* \Psi_G$
and $A_G \sharp* \Psi$
and $A_G \sharp* \Psi_F$
and $A_F \sharp* A_G$
and $A_F \sharp* P$
and $A_G \sharp* P$
and $A_P \sharp* A_F$
and $A_P \sharp* A_G$
and $A_P \sharp* \Psi_F$

and $A_P \sharp* \Psi_G$
and $A_P \sharp* P$

shows $\Psi_G \rhd P \Longrightarrow\!\hat{}_\tau\ P'$
⟨*proof*⟩

coinductive *quiet* :: $('a, 'b, 'c)$ *psi* $\Rightarrow$ *bool*
where $[\![\forall \Psi.\ (\mathit{insertAssertion}\ (\mathit{extractFrame}\ P)\ \Psi \simeq_F \langle \varepsilon, \Psi\rangle \wedge$
$(\forall Rs.\ \Psi \rhd P \longmapsto Rs \longrightarrow (\exists P'.\ Rs = \tau \prec P' \wedge \mathit{quiet}\ P')))]\!] \Longrightarrow \mathit{quiet}$
P

lemma *quietFrame*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*

assumes *quiet* P

shows *insertAssertion* (*extractFrame* P) $\Psi \simeq_F \langle \varepsilon, \Psi\rangle$
⟨*proof*⟩

lemma *quietTransition*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Rs :: $('a, 'b, 'c)$ *residual*

assumes *quiet* P
and $\Psi \rhd P \longmapsto Rs$

obtains P' **where** $Rs = \tau \prec P'$ **and** *quiet* P'
⟨*proof*⟩

lemma *quietEqvt*:
fixes P :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*

assumes *quiet* P

shows *quiet*$(p \cdot P)$
⟨*proof*⟩

lemma *quietOutput*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$
and *quiet* P

shows *False*
⟨*proof*⟩

lemma *quietInput*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \longmapsto M(\!|N|\!) \prec P'$
and *quiet* P

shows *False*
⟨*proof*⟩

lemma *quietTau*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \longmapsto \tau \prec P'$
and *insertAssertion* (*extractFrame* P) $\Psi \simeq_F \langle \varepsilon, \Psi\rangle$
and *quiet* P

shows *quiet* P'
⟨*proof*⟩

lemma *tauChainCases*[*consumes 1*, *case-names TauBase TauStep*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \Longrightarrow\!\hat{}_\tau\ P'$
and $P = P' \Longrightarrow Prop$
and $\Psi \rhd P \Longrightarrow_\tau P' \Longrightarrow Prop$

shows *Prop*
⟨*proof*⟩

end

end

theory *Weak-Simulation*
imports *Simulation Tau-Chain*

begin

context *env* **begin**

definition
weakSimulation :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow$
$('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set \Rightarrow$
$('a, 'b, 'c)\ psi \Rightarrow bool$ ($\langle$- $\triangleright$ - $\rightsquigarrow$<-> -$\rangle$ [80, 80, 80, 80] 80)
where
$\Psi \triangleright P \rightsquigarrow <Rel> Q \equiv (\forall \Psi'\ \alpha\ Q'.\ \Psi \triangleright Q \longmapsto \alpha \prec Q' \longrightarrow bn\ \alpha\ \sharp*\ \Psi \longrightarrow bn\ \alpha\ \sharp*\ P \longrightarrow \alpha \neq \tau \longrightarrow$
$(\exists P''.\ \Psi : Q \triangleright P \Longrightarrow \alpha \prec P'' \wedge (\exists P'.\ \Psi \otimes \Psi' \triangleright P'' \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi \otimes \Psi', P', Q') \in Rel))) \wedge$
$(\forall Q'.\ \Psi \triangleright Q \longmapsto \tau \prec Q' \longrightarrow (\exists P'.\ \Psi \triangleright P \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi, P', Q') \in Rel))$

abbreviation
weakSimulationNilJudge ($\langle$- $\rightsquigarrow$<-> -$\rangle$ [80, 80, 80] 80) **where** $P \rightsquigarrow <Rel> Q \equiv SBottom' \triangleright P \rightsquigarrow <Rel> Q$

lemma *weakSimI*[*consumes 1*, *case-names cAct cTau*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and *rAct*: $\bigwedge \Psi'\ \alpha\ Q'.\ [\![\Psi \triangleright Q \longmapsto \alpha \prec Q';\ bn\ \alpha\ \sharp*\ \Psi;\ bn\ \alpha\ \sharp*\ P;\ bn\ \alpha\ \sharp*\ Q;$
$bn\ \alpha\ \sharp*\ subject\ \alpha;\ bn\ \alpha\ \sharp*\ C;\ distinct(bn\ \alpha);\ \alpha \neq \tau]\!] \Longrightarrow$

$\exists P''.\ \Psi : Q \triangleright P \Longrightarrow \alpha \prec P'' \wedge (\exists P'.\ \Psi \otimes \Psi' \triangleright P'' \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi \otimes \Psi', P', Q') \in Rel)$
and *rTau*: $\bigwedge Q'.\ \Psi \triangleright Q \longmapsto \tau \prec Q' \Longrightarrow \exists P'.\ \Psi \triangleright P \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi, P', Q') \in Rel$

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

lemma *weakSimI2*[*case-names cAct cTau*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and C :: $'d$::*fs-name*

assumes *rOutput*: $\bigwedge \Psi'\ \alpha\ Q'.\ [\![\Psi \triangleright Q \longmapsto \alpha \prec Q';\ bn\ \alpha\ \sharp*\ \Psi;\ bn\ \alpha\ \sharp*\ P;\ \alpha \neq \tau]\!] \Longrightarrow$
$\exists P''.\ \Psi : Q \triangleright P \Longrightarrow \alpha \prec P'' \wedge (\exists P'.\ \Psi \otimes \Psi' \triangleright P''$

$\Longrightarrow\hat{}_\tau\ P' \wedge (\Psi \otimes \Psi',\ P',\ Q') \in Rel$)
and *rTau*: $\bigwedge Q'.\ \Psi \rhd Q \longmapsto \tau \prec Q' \Longrightarrow \exists\, P'.\ \Psi \rhd P \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi,\ P',\ Q') \in Rel$

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

lemma *weakSimIChainFresh*[*consumes 4*, *case-names cOutput cInput*]:
fixes Ψ :: $'b$
and P :: $('a,\ 'b,\ 'c)\ psi$
and Rel :: $('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and Q :: $('a,\ 'b,\ 'c)\ psi$
and $yvec$ :: *name list*
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and $yvec \sharp* \Psi$
and $yvec \sharp* P$
and $yvec \sharp* Q$
and *rAct*: $\bigwedge \Psi'\ \alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto \alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ \alpha \neq \tau;$
$bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ yvec \sharp* \alpha;\ yvec \sharp* Q';\ yvec \sharp* \Psi']\!] \Longrightarrow$
$\exists\, P''.\ \Psi : Q \rhd P \Longrightarrow \alpha \prec P'' \wedge (\exists\, P'.\ \Psi \otimes \Psi' \rhd P'' \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi \otimes \Psi',\ P',\ Q') \in Rel)$
and *rTau*: $\bigwedge Q'.\ [\![\Psi \rhd Q \longmapsto \tau \prec Q';\ yvec \sharp* Q']\!] \Longrightarrow \exists\, P'.\ \Psi \rhd P \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi,\ P',\ Q') \in Rel$

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

lemma *weakSimIFresh*[*consumes 4*, *case-names cAct cTau*]:
fixes Ψ :: $'b$
and P :: $('a,\ 'b,\ 'c)\ psi$
and Rel :: $('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and Q :: $('a,\ 'b,\ 'c)\ psi$
and x :: *name*
and C :: $'d$::*fs-name*

assumes *Eqvt*: *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp P$
and $x \sharp Q$
and $\bigwedge \alpha\ Q'\ \Psi'.\ [\![\Psi \rhd Q \longmapsto \alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P;\ bn\ \alpha \sharp* Q;\ \alpha \neq \tau;$
$bn\ \alpha \sharp* subject\ \alpha;\ bn\ \alpha \sharp* C;\ x \sharp \alpha;\ x \sharp Q';\ x \sharp \Psi']\!] \Longrightarrow$
$\exists\, P''.\ \Psi : Q \rhd P \Longrightarrow \alpha \prec P'' \wedge (\exists\, P'.\ \Psi \otimes \Psi' \rhd P'' \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi \otimes \Psi',\ P',\ Q') \in Rel)$
and $\bigwedge Q'.\ [\![\Psi \rhd Q \longmapsto \tau \prec Q';\ x \sharp Q']\!] \Longrightarrow \exists\, P'.\ \Psi \rhd P \Longrightarrow\hat{}_\tau\ P' \wedge (\Psi,\ P',\ Q') \in Rel$

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

lemma *weakSimE*:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \rightsquigarrow <Rel> Q$

shows $\bigwedge \Psi' \ \alpha \ Q'. [\![\Psi \rhd Q \longmapsto \alpha \prec Q'; bn\ \alpha \sharp* \Psi; bn\ \alpha \sharp* P; \alpha \neq \tau]\!] \Longrightarrow$
$\exists P''. \Psi : Q \rhd P \Longrightarrow \alpha \prec P'' \wedge (\exists P'. \Psi \otimes \Psi' \rhd P'' \Longrightarrow\hat{}_\tau$
$P' \wedge (\Psi \otimes \Psi', P', Q') \in Rel)$
and $\bigwedge Q'. \Psi \rhd Q \longmapsto \tau \prec Q' \Longrightarrow \exists P'. \Psi \rhd P \Longrightarrow\hat{}_\tau P' \wedge (\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *weakSimClosedAux*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes *EqvtRel*: *eqvt Rel*
and *PSimQ*: $\Psi \rhd P \rightsquigarrow <Rel> Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow <Rel> (p \cdot Q)$
⟨*proof*⟩

lemma *weakSimClosed*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes *EqvtRel*: *eqvt Rel*

shows $\Psi \rhd P \rightsquigarrow <Rel> Q \Longrightarrow (p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow <Rel> (p \cdot Q)$
and $P \rightsquigarrow <Rel> Q \Longrightarrow (p \cdot P) \rightsquigarrow <Rel> (p \cdot Q)$
⟨*proof*⟩

lemma *weakSimReflexive*:
fixes $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes $\{(\Psi, P, P) \mid \Psi\ P.\ True\} \subseteq Rel$

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

lemma *weakSimTauChain*:
fixes Ψ :: $'b$
and Q :: $('a, 'b, 'c)\ psi$
and Q' :: $('a, 'b, 'c)\ psi$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

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

obtains P' **where** $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P'$ **and** $(\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *weakSimE2*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$

assumes *PRelQ*: $(\Psi, P, Q) \in Rel$
and *Sim*: $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in Rel \Longrightarrow \Psi' \rhd R \rightsquigarrow <Rel> S$
and *QTrans*: $\Psi : R \rhd Q \Longrightarrow \alpha \prec Q'$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* P$
and $\alpha \neq \tau$

obtains $P''\ P'$ **where** $\Psi : R \rhd P \Longrightarrow \alpha \prec P''$ **and** $\Psi \otimes \Psi' \rhd P'' \Longrightarrow\hat{}_\tau\ P'$ **and** $(\Psi \otimes \Psi', P', Q') \in Rel$
⟨*proof*⟩

lemma *weakSimTransitive*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and T :: $('a, 'b, 'c)\ psi$
and Rel'' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PRelQ*: $(\Psi, P, Q) \in Rel$
and *QSimR*: $\Psi \rhd Q \rightsquigarrow <Rel'> R$
and *Eqvt*: *eqvt* Rel''
and *Set*: $\{(\Psi, P, R) \mid \Psi\ P\ R.\ \exists Q.\ (\Psi, P, Q) \in Rel \wedge (\Psi, Q, R) \in Rel'\} \subseteq$

Rel''
and *Sim*: $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in Rel \Longrightarrow \Psi' \rhd R \rightsquigarrow <Rel> S$

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

lemma *weakSimStatEq*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and Rel :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*
and Q :: *('a, 'b, 'c) psi*
and Ψ' :: *'b*

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow <Rel> Q$
and *eqvt Rel'*
and $\Psi \simeq \Psi'$
and *C1*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ [\![(\Psi', R, S) \in Rel; \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', R, S) \in Rel'$

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

lemma *weakSimMonotonic*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and A :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*
and Q :: *('a, 'b, 'c) psi*
and B :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*

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

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

lemma *strongSimWeakSim*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and Q :: *('a, 'b, 'c) psi*
and Rel :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*

assumes *PRelQ*: $(\Psi, P, Q) \in Rel$
and *StatImp*: $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in Rel \Longrightarrow insertAssertion(extractFrame\ S)\ \Psi' \hookrightarrow_F insertAssertion(extractFrame\ R)\ \Psi'$
and *Sim*: $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in Rel \Longrightarrow \Psi' \rhd R \rightsquigarrow [Rel]\ S$
and *Ext*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi', R, S) \in Rel \Longrightarrow (\Psi' \otimes \Psi'', R, S) \in Rel$

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

lemma *strongAppend*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*
and Q :: *('a, 'b, 'c) psi*
and R :: *('a, 'b, 'c) psi*
and *Rel* :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*
and *Rel'* :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*
and *Rel''* :: *('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set*

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow <Rel> Q$
and *QSimR*: $\Psi \rhd Q \rightsquigarrow [Rel'] R$
and *Eqvt''*: *eqvt Rel''*
and *RimpQ*: *insertAssertion (extractFrame R) Ψ ↪F insertAssertion (extractFrame Q) Ψ*
and *Set*: $\{(\Psi, P, R) \mid \Psi\ P\ R.\ \exists Q.\ (\Psi, P, Q) \in Rel \wedge (\Psi, Q, R) \in Rel'\} \subseteq Rel''$
and *C1*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in Rel' \Longrightarrow (\Psi \otimes \Psi', P, Q) \in Rel'$

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

lemma *quietSim*:
fixes Ψ :: *'b*
and P :: *('a, 'b, 'c) psi*

assumes *quiet P*
and *eqvt Rel*
and *cQuiet*: $\bigwedge P.$ *quiet P* $\Longrightarrow (\Psi, \mathbf{0}, P) \in Rel$

shows $\Psi \rhd \mathbf{0} \rightsquigarrow <Rel> P$
⟨*proof*⟩

end

end

theory *Weak-Stat-Imp*
imports *Tau-Chain*
begin

context *env* **begin**

definition
weakStatImp :: *'b ⇒ ('a, 'b, 'c) psi ⇒*
('b × ('a, 'b, 'c) psi × ('a, 'b, 'c) psi) set ⇒
('a, 'b, 'c) psi ⇒ bool (‹- ⊳ - ≲<-> -› [80, 80, 80, 80] 80)
where $\Psi \rhd P \lesssim <Rel> Q \equiv \forall \Psi'.\ \exists Q'\ Q''.\ \Psi \rhd Q \Longrightarrow\hat{}_{\tau}\ Q' \wedge$ *insertAssertion(extractFrame P) Ψ ↪F insertAssertion(extractFrame Q') Ψ ∧ Ψ ⊗ Ψ' ⊳ Q'*

$\Longrightarrow\hat{}_\tau\ Q'' \wedge (\Psi \otimes \Psi',\ P,\ Q'') \in Rel$

lemma *weakStatImpMonotonic*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $A :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $B :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes $\Psi \rhd P \lessapprox <A> Q$
and $A \subseteq B$

shows $\Psi \rhd P \lessapprox <B> Q$
⟨*proof*⟩

lemma *weakStatImpI*[*case-names cStatImp*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Rel :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $\Psi' :: 'b$

assumes $\bigwedge \Psi'.\ \exists\, Q'\ Q''.\ \Psi \rhd Q \Longrightarrow\hat{}_\tau\ Q' \wedge insertAssertion(extractFrame\ P)\ \Psi \hookrightarrow_F insertAssertion(extractFrame\ Q')\ \Psi \wedge \Psi \otimes \Psi' \rhd Q' \Longrightarrow\hat{}_\tau\ Q'' \wedge (\Psi \otimes \Psi',\ P,\ Q'') \in Rel$

shows $\Psi \rhd P \lessapprox <Rel> Q$
⟨*proof*⟩

lemma *weakStatImpE*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Rel :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \lessapprox <Rel> Q$

obtains $Q'\ Q''$ **where** $\Psi \rhd Q \Longrightarrow\hat{}_\tau\ Q'$ **and** $insertAssertion(extractFrame\ P)\ \Psi \hookrightarrow_F insertAssertion(extractFrame\ Q')\ \Psi$ **and** $\Psi \otimes \Psi' \rhd Q' \Longrightarrow\hat{}_\tau\ Q''$ **and** $(\Psi \otimes \Psi',\ P,\ Q'') \in Rel$

⟨*proof*⟩

lemma *weakStatImpClosed*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Rel :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and $Q :: ('a,\ 'b,\ 'c)\ psi$

and p :: *name prm*

assumes *EqvtRel*: *eqvt Rel*
and *PStatImpQ*: $\Psi \rhd P \lessapprox <Rel> Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \lessapprox <Rel> (p \cdot Q)$
⟨*proof*⟩

lemma *weakStatImpReflexive*:
fixes *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

assumes $\{(\Psi, P, P) \mid \Psi\ P.\ True\} \subseteq Rel$

shows $\Psi \rhd P \lessapprox <Rel> P$
⟨*proof*⟩

lemma *weakStatImpTransitive*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and *Rel'* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and R :: $('a, 'b, 'c)\ psi$
and *Rel''* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PStatImpQ*: $\Psi \rhd P \lessapprox <Rel> Q$
and *QRelR*: $(\Psi, Q, R) \in Rel'$
and *Set*: $\{(\Psi', S, U) \mid \Psi'\ S\ U.\ \exists T.\ (\Psi', S, T) \in Rel \wedge (\Psi', T, U) \in Rel'\}$
$\subseteq Rel''$
and *C1*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel' \Longrightarrow \Psi' \rhd S \lessapprox <Rel'> T$
and *C2*: $\bigwedge \Psi'\ S\ T\ S'.\ [\![(\Psi', S, T) \in Rel'; \Psi' \rhd S \Longrightarrow\!\hat{}_{\tau}\ S']\!] \Longrightarrow \exists T'.\ \Psi' \rhd$
$T \Longrightarrow\!\hat{}_{\tau}\ T' \wedge (\Psi', S', T') \in Rel'$

shows $\Psi \rhd P \lessapprox <Rel''> R$
⟨*proof*⟩

lemma *weakStatImpStatEq*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and Ψ' :: $'b$

assumes *PSimQ*: $\Psi \rhd P \lessapprox <Rel> Q$
and $\Psi \simeq \Psi'$
and *C1*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ [\![(\Psi', R, S) \in Rel; \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', R, S) \in Rel'$

shows $\Psi' \rhd P \lesssim_{<Rel'>} Q$
⟨*proof*⟩

lemma *statImpWeakStatImp*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PImpQ*: *insertAssertion*(*extractFrame* P) $\Psi \hookrightarrow_F$ *insertAssertion*(*extractFrame* Q) Ψ
and *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', P, Q) \in Rel$

shows $\Psi \rhd P \lesssim_{<Rel>} Q$
⟨*proof*⟩

end

end

theory *Bisimulation*
imports *Simulation*
begin

context *env* **begin**

lemma *monoCoinduct*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \rightsquigarrow[\{(xc, xb, xa).\ x\ xc\ xb\ xa\}]\ P) \longrightarrow$
$(\Psi \rhd Q \rightsquigarrow[\{(xb, xa, xc).\ y\ xb\ xa\ xc\}]\ P)$
⟨*proof*⟩

coinductive-set *bisim* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
where
step: $[\![(insertAssertion\ (extractFrame\ P))\ \Psi \simeq_F (insertAssertion\ (extractFrame\ Q)\ \Psi);$
$\Psi \rhd P \rightsquigarrow[bisim]\ Q;$
$\forall \Psi'.\ (\Psi \otimes \Psi',\ P, Q) \in bisim; (\Psi, Q, P) \in bisim]\!] \Longrightarrow (\Psi, P, Q) \in bisim$
monos *monoCoinduct*

abbreviation
bisimJudge (‹- ⊳ - ∼ -› [70, 70, 70] 65) **where** $\Psi \rhd P \sim Q \equiv (\Psi, P, Q) \in bisim$
abbreviation
bisimNilJudge (‹- ∼ -› [70, 70] 65) **where** $P \sim Q \equiv SBottom' \rhd P \sim Q$

lemma *bisimCoinductAux*[*consumes 1*]:
fixes F :: $'b$
and P :: $('a, 'b, 'c)$ *psi*

and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi \wedge$
$(\Psi \rhd P \rightsquigarrow [(X \cup bisim)]\ Q) \wedge$
$(\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X \vee (\Psi \otimes \Psi', P, Q) \in bisim) \wedge$
$((\Psi, Q, P) \in X \vee (\Psi, Q, P) \in bisim)$

shows $(\Psi, P, Q) \in bisim$
⟨*proof*⟩

lemma *bisimCoinduct*[*consumes 1*, *case-names cStatEq cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow insertAssertion\ (extractFrame\ R)\ \Psi' \simeq_F insertAssertion\ (extractFrame\ S)\ \Psi'$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow [(X \cup bisim)]\ S$
and $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi', R, S) \in X \Longrightarrow (\Psi' \otimes \Psi'', R, S) \in X \vee (\Psi' \otimes \Psi'', R, S) \in bisim$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow (\Psi', S, R) \in X \vee (\Psi', S, R) \in bisim$

shows $(\Psi, P, Q) \in bisim$
⟨*proof*⟩

lemma *bisimWeakCoinductAux*[*consumes 1*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi \wedge$
$\Psi \rhd P \rightsquigarrow [X]\ Q \wedge$
$(\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X) \wedge (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in bisim$
⟨*proof*⟩

lemma *bisimWeakCoinduct*[*consumes 1*, *case-names cStatEq cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow [X]\ Q$
and $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in bisim$
⟨*proof*⟩

lemma *bisimE*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$
and $\Psi' :: 'b$

assumes $(\Psi, P, Q) \in bisim$

shows $insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$
and $\Psi \rhd P \rightsquigarrow [bisim]\ Q$
and $(\Psi \otimes \Psi', P, Q) \in bisim$
and $(\Psi, Q, P) \in bisim$
⟨*proof*⟩

lemma *bisimI*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$

assumes $insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$
and $\Psi \rhd P \rightsquigarrow [bisim]\ Q$
and $\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in bisim$
and $(\Psi, Q, P) \in bisim$

shows $(\Psi, P, Q) \in bisim$
⟨*proof*⟩

lemma *bisimReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \sim P$
⟨*proof*⟩

lemma *bisimClosed*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes *PBisimQ*: $\Psi \rhd P \sim Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \sim (p \cdot Q)$
⟨*proof*⟩

lemma *bisimEqvt*[*simp*]:
shows *eqvt bisim*
⟨*proof*⟩

lemma *statEqBisim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \sim Q$
and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \sim Q$
⟨*proof*⟩

lemma *bisimTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes *PQ*: $\Psi \rhd P \sim Q$
and *QR*: $\Psi \rhd Q \sim R$

shows $\Psi \rhd P \sim R$
⟨*proof*⟩

lemma *weakTransitiveCoinduct*[*case-names cStatEq cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *p*: $(\Psi, P, Q) \in X$
and *Eqvt*: *eqvt X*

and *rStatEq*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \mathit{insertAssertion}\ (\mathit{extractFrame}\ P)\ \Psi \simeq_F \mathit{insertAssertion}\ (\mathit{extractFrame}\ Q)\ \Psi$
and *rSim*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow[(\{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\})]\ Q$
and *rExt*: $\bigwedge\Psi\ P\ Q\ \Psi'.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi \otimes \Psi',\ P,\ Q) \in X$
and *rSym*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi,\ Q,\ P) \in X$

shows $\Psi \rhd P \sim Q$
⟨*proof*⟩

lemma *weakTransitiveCoinduct'*[*case-names cStatEq cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *p*: $(\Psi,\ P,\ Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatEq*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \mathit{insertAssertion}\ (\mathit{extractFrame}\ P)\ \Psi \simeq_F \mathit{insertAssertion}\ (\mathit{extractFrame}\ Q)\ \Psi$
and *rSim*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow[(\{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\})]\ Q$
and *rExt*: $\bigwedge\Psi\ P\ Q\ \Psi'.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi \otimes \Psi',\ P,\ Q) \in X$
and *rSym*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi,\ Q,\ P) \in \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\}$

shows $\Psi \rhd P \sim Q$
⟨*proof*⟩

lemma *weakTransitiveCoinduct''*[*case-names cStatEq cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *p*: $(\Psi,\ P,\ Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatEq*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \mathit{insertAssertion}\ (\mathit{extractFrame}\ P)\ \Psi \simeq_F \mathit{insertAssertion}\ (\mathit{extractFrame}\ Q)\ \Psi$
and *rSim*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow[(\{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge$

$\Psi \rhd Q' \sim Q\})]\ Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi,\ P,\ Q) \in \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\} \Longrightarrow (\Psi \otimes \Psi',\ P,\ Q) \in \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\}$
and *rSym*: $\bigwedge \Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\} \Longrightarrow (\Psi,\ Q,\ P) \in \{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in X \wedge \Psi \rhd Q' \sim Q\}$

shows $\Psi \rhd P \sim Q$
⟨*proof*⟩

lemma *transitiveCoinduct*[*case-names cStatEq cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *p*: $(\Psi,\ P,\ Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatEq*: $\bigwedge \Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow insertAssertion\ (extractFrame\ R)\ \Psi' \simeq_F insertAssertion\ (extractFrame\ S)\ \Psi'$
and *rSim*: $\bigwedge \Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow[(\{(\Psi',\ R,\ S) \mid \Psi'\ R\ R'\ S'\ S.\ \Psi' \rhd R \sim R' \wedge ((\Psi',\ R',\ S') \in X \vee \Psi' \rhd R' \sim S') \wedge \Psi' \rhd S' \sim S\})]\ S$
and *rExt*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi',\ R,\ S) \in X \Longrightarrow (\Psi' \otimes \Psi'',\ R,\ S) \in X \vee \Psi' \otimes \Psi'' \rhd R \sim S$
and *rSym*: $\bigwedge \Psi'\ R\ S.\ (\Psi',\ R,\ S) \in X \Longrightarrow (\Psi',\ S,\ R) \in X \vee \Psi' \rhd S \sim R$

shows $\Psi \rhd P \sim Q$
⟨*proof*⟩

lemma *transitiveCoinduct'*[*case-names cStatEq cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *p*: $(\Psi,\ P,\ Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatEq*: $\bigwedge \Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow insertAssertion\ (extractFrame\ P)\ \Psi \simeq_F insertAssertion\ (extractFrame\ Q)\ \Psi$
and *rSim*: $\bigwedge \Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow[(\{(\Psi,\ P,\ Q) \mid \Psi\ P\ P'\ Q'\ Q.$

$\Psi \rhd P \sim P' \wedge$
$(\Psi, P', Q') \in (X \cup bisim) \wedge$
$\Psi \rhd Q' \sim Q\})] Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X \vee \Psi \otimes \Psi' \rhd P \sim Q$
and *rSym*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow$
$(\Psi, Q, P) \in \{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge ((\Psi, P', Q') \in (X \cup bisim)) \wedge \Psi \rhd Q' \sim Q\}$

shows $\Psi \rhd P \sim Q$
⟨*proof*⟩

lemma *bisimSymmetric*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd Q \sim P$
⟨*proof*⟩

lemma *eqvtTrans*[*intro*]:
assumes *eqvt X*

shows *eqvt* $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge ((\Psi, P', Q') \in X \vee \Psi \rhd P' \sim Q') \wedge \Psi \rhd Q' \sim Q\}$
⟨*proof*⟩

lemma *eqvtWeakTrans*[*intro*]:
assumes *eqvt X*

shows *eqvt* $\{(\Psi, P, Q) \mid \Psi\ P\ P'\ Q'\ Q.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\}$
⟨*proof*⟩

end

end

theory *Sim-Pres*
imports *Simulation*
begin

context *env* **begin**

lemma *inputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$

assumes $PRelQ$: $\bigwedge Tvec.\ length\ xvec = length\ Tvec \Longrightarrow (\Psi, P[xvec::=Tvec], Q[xvec::=Tvec]) \in Rel$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \rightsquigarrow[Rel]\ M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *outputPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes $PRelQ$: $(\Psi, P, Q) \in Rel$

shows $\Psi \rhd M\langle N\rangle.P \rightsquigarrow[Rel]\ M\langle N\rangle.Q$
⟨*proof*⟩

lemma *casePres*:
fixes $\Psi :: 'b$
and $CsP :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $CsQ :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $M :: 'a$
and $N :: 'a$

assumes $PRelQ$: $\bigwedge \varphi\ Q.\ (\varphi, Q)\ mem\ CsQ \Longrightarrow \exists P.\ (\varphi, P)\ mem\ CsP \wedge guarded\ P \wedge (\Psi, P, Q) \in Rel$
and Sim: $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in Rel \Longrightarrow \Psi' \rhd R \rightsquigarrow[Rel]\ S$
and $Rel \subseteq Rel'$

shows $\Psi \rhd Cases\ CsP \rightsquigarrow[Rel']\ Cases\ CsQ$
⟨*proof*⟩

lemma *resPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $Rel' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
and *eqvt Rel'*
and $x \sharp \Psi$
and $Rel \subseteq Rel'$
and *C1*: $\bigwedge \Psi'\ R\ S\ y.\ [\![(\Psi', R, S) \in Rel; y \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)R, (\!|\nu y|\!)S) \in Rel'$

shows $\Psi \rhd (\!|\nu x|\!)P \rightsquigarrow[Rel'] (\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *resChainPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and *xvec* :: *name list*

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow[Rel]\ Q$
and *eqvt Rel*
and $xvec \sharp* \Psi$
and *C1*: $\bigwedge \Psi'\ R\ S\ y.\ [\![(\Psi', R, S) \in Rel; y \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)R, (\!|\nu y|\!)S) \in Rel$

shows $\Psi \rhd (\!|\nu* xvec|\!)P \rightsquigarrow[Rel] (\!|\nu* xvec|\!)Q$
⟨*proof*⟩

lemma *parPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PRelQ*: $\bigwedge A_R\ \Psi_R.\ [\![extractFrame\ R = \langle A_R, \Psi_R\rangle; A_R \sharp* \Psi; A_R \sharp* P; A_R \sharp* Q]\!] \Longrightarrow (\Psi \otimes \Psi_R, P, Q) \in Rel$
and *Eqvt*: *eqvt Rel*
and *Eqvt'*: *eqvt Rel'*

and *StatImp*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow insertAssertion\ (extractFrame\ T)\ \Psi' \hookrightarrow_F insertAssertion\ (extractFrame\ S)\ \Psi'$
and *Sim*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow \Psi' \rhd S \rightsquigarrow[Rel]\ T$
and *Ext*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel]\!] \Longrightarrow (\Psi' \otimes \Psi'', S, T) \in Rel$

and *C1*: $\bigwedge \Psi'\ S\ T\ A_U\ \Psi_U\ U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel; extractFrame\ U = \langle A_U, \Psi_U\rangle; A_U \sharp* \Psi'; A_U \sharp* S; A_U \sharp* T]\!] \Longrightarrow (\Psi', S \parallel U, T \parallel U) \in Rel'$
and *C2*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel'; xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu* xvec|\!)S, (\!|\nu* xvec|\!)T) \in Rel'$

and *C3*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel; \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', S, T) \in Rel$

shows $\Psi \rhd P \parallel R \rightsquigarrow [Rel']\ Q \parallel R$
⟨*proof*⟩
unbundle *no relcomp-syntax*
lemma *bangPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in Rel$
and *eqvt Rel'*
and *guarded P*
and *guarded Q*
and *cSim*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow \Psi' \rhd S \rightsquigarrow [Rel]\ T$
and *cExt*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ (\Psi', S, T) \in Rel \Longrightarrow (\Psi' \otimes \Psi'', S, T) \in Rel$
and *cSym*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow (\Psi', T, S) \in Rel$
and *StatEq*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel; \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', S, T) \in Rel$
and *Closed*: $\bigwedge \Psi'\ S\ T\ p.\ (\Psi', S, T) \in Rel \Longrightarrow ((p::name\ prm) \cdot \Psi', p \cdot S, p \cdot T) \in Rel$
and *Assoc*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi', S \parallel (T \parallel U), (S \parallel T) \parallel U) \in Rel$
and *ParPres*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi', S, T) \in Rel \Longrightarrow (\Psi', S \parallel U, T \parallel U) \in Rel$
and *FrameParPres*: $\bigwedge \Psi'\ \Psi_U\ S\ T\ U\ A_U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel; extractFrame\ U = \langle A_U, \Psi_U\rangle; A_U \sharp* \Psi'; A_U \sharp* S; A_U \sharp* T]\!] \Longrightarrow (\Psi', U \parallel S, U \parallel T) \in Rel$
and *ResPres*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel; xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu*xvec|\!)S, (\!|\nu*xvec|\!)\,T) \in Rel$
and *ScopeExt*: $\bigwedge xvec\ \Psi'\ S\ T.\ [\![xvec \sharp* \Psi'; xvec \sharp* T]\!] \Longrightarrow (\Psi', (\!|\nu*xvec|\!)(S \parallel T), ((\!|\nu*xvec|\!)S) \parallel T) \in Rel$
and *Trans*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi', S, T) \in Rel; (\Psi', T, U) \in Rel]\!] \Longrightarrow (\Psi', S, U) \in Rel$
and *Compose*: $\bigwedge \Psi'\ S\ T\ U\ O.\ [\![(\Psi', S, T) \in Rel; (\Psi', T, U) \in Rel'; (\Psi', U, O) \in Rel]\!] \Longrightarrow (\Psi', S, O) \in Rel'$
and *C1*: $\bigwedge \Psi\ S\ T\ U.\ [\![(\Psi, S, T) \in Rel; guarded\ S; guarded\ T]\!] \Longrightarrow (\Psi, U \parallel !S, U \parallel !T) \in Rel'$
and *Der*: $\bigwedge \Psi'\ S\ \alpha\ S'\ T.\ [\![\Psi' \rhd !S \longmapsto \alpha \prec S'; (\Psi', S, T) \in Rel; bn\ \alpha \sharp* \Psi'; bn\ \alpha \sharp* S; bn\ \alpha \sharp* T; guarded\ T; bn\ \alpha \sharp* subject\ \alpha]\!] \Longrightarrow \exists\, T'\ U\ O.\ \ \Psi' \rhd !T \longmapsto \alpha \prec T' \wedge (\Psi', S', U \parallel !S) \in Rel \wedge (\Psi', T', O \parallel !T) \in Rel \wedge (\Psi', U, O) \in Rel \wedge ((supp\ U)::name\ set) \subseteq supp\ S' \wedge ((supp\ O)::name\ set) \subseteq supp\ T'$

shows $\Psi \rhd R \parallel !P \rightsquigarrow [Rel']\ R \parallel !Q$
⟨*proof*⟩

unbundle *relcomp-syntax*
end

end

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

context *env* **begin**

lemma *bisimInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $\bigwedge Tvec.$ *length xvec* = *length Tvec* $\Longrightarrow \Psi \rhd P[xvec::=Tvec] \sim Q[xvec::=Tvec]$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \sim M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *bisimOutputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd M\langle N\rangle.P \sim M\langle N\rangle.Q$
⟨*proof*⟩

lemma *bisimCasePres*:
fixes Ψ :: $'b$
and *CsP* :: $('c \times ('a, 'b, 'c)$ *psi*$)$ *list*
and *CsQ* :: $('c \times ('a, 'b, 'c)$ *psi*$)$ *list*

assumes $\bigwedge \varphi\ P.\ (\varphi, P)$ *mem CsP* $\Longrightarrow \exists\, Q.\ (\varphi, Q)$ *mem CsQ* $\wedge$ *guarded Q* $\wedge\ \Psi \rhd P \sim Q$
and $\bigwedge \varphi\ Q.\ (\varphi, Q)$ *mem CsQ* $\Longrightarrow \exists\, P.\ (\varphi, P)$ *mem CsP* $\wedge$ *guarded P* $\wedge\ \Psi \rhd P \sim Q$

shows $\Psi \rhd$ *Cases CsP* $\sim$ *Cases CsQ*
⟨*proof*⟩

lemma *bisimResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \sim Q$
and $x \sharp \Psi$

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

lemma *bisimResChainPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $\Psi \rhd P \sim Q$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu * xvec|\!) P \sim (\!|\nu * xvec|\!) Q$
⟨*proof*⟩

lemma *bisimParPresAux*:
fixes $\Psi :: 'b$
and $\Psi_R :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$
and $A_R :: name\ list$

assumes $\Psi \otimes \Psi_R \rhd P \sim Q$
and *FrR*: $extractFrame\ R = \langle A_R, \Psi_R \rangle$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$

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

lemma *bisimParPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim Q$

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

end

end

theory *Sim-Struct-Cong*
imports *Simulation*
begin

lemma *partitionListLeft*:
assumes $xs@ys{=}xs'@y\#ys'$
and y *mem* xs
and *distinct*$(xs@ys)$

obtains zs **where** $xs = xs'@y\#zs$ **and** $ys'{=}zs@ys$
⟨*proof*⟩

lemma *partitionListRight*:
assumes $xs@ys{=}xs'@y\#ys'$
and y *mem* ys
and *distinct*$(xs@ys)$

obtains zs **where** $xs' = xs@zs$ **and** $ys{=}zs@y\#ys'$
⟨*proof*⟩

context *env* **begin**

lemma *resComm*:
fixes Ψ :: $'b$
and x :: *name*
and y :: *name*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and P :: $('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $y \sharp \Psi$
and *eqvt Rel*
and *R1*: $\bigwedge \Psi'\ Q.\ (\Psi', Q, Q) \in Rel$
and *R2*: $\bigwedge \Psi'\ a\ b\ Q.\ [\![a \sharp \Psi'; b \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu a|\!)((\!|\nu b|\!)Q), (\!|\nu b|\!)((\!|\nu a|\!)Q)) \in Rel$

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

lemma *parAssocLeft*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *eqvt Rel*
and *C1*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi, (S \parallel T) \parallel U, S \parallel (T \parallel U)) \in Rel$
and *C2*: $\bigwedge xvec\ \Psi'\ S\ T\ U.\ [\![xvec \sharp* \Psi'; xvec \sharp* S]\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!)((S \parallel T) \parallel U), S \parallel ((\!|\nu * xvec|\!)(T \parallel U))) \in Rel$
and *C3*: $\bigwedge xvec\ \Psi'\ S\ T\ U.\ [\![xvec \sharp* \Psi'; xvec \sharp* U]\!] \Longrightarrow (\Psi', ((\!|\nu * xvec|\!)(S \parallel T)) \parallel U, (\!|\nu * xvec|\!)(S \parallel (T \parallel U))) \in Rel$
and *C4*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel; xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!)S, (\!|\nu * xvec|\!)T) \in Rel$

shows $\Psi \rhd (P \parallel Q) \parallel R \rightsquigarrow [Rel]\ P \parallel (Q \parallel R)$
⟨*proof*⟩

lemma *parNilLeft*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *eqvt Rel*
and *C1*: $\bigwedge Q.\ (\Psi, Q \parallel \mathbf{0}, Q) \in Rel$

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

lemma *parNilRight*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *eqvt Rel*
and *C1*: $\bigwedge Q.\ (\Psi, Q, (Q \parallel \mathbf{0})) \in Rel$

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

lemma *resNilLeft*:
fixes x :: *name*
and Ψ :: $'b$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

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

lemma *resNilRight*:
fixes x :: *name*
and Ψ :: $'b$

and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

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

lemma *inputPushResLeft*:
fixes $\Psi :: 'b$
and $x :: name$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$
and $C1: \bigwedge Q.\ (\Psi, Q, Q) \in Rel$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \rightsquigarrow[Rel]\ M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *inputPushResRight*:
fixes $\Psi :: 'b$
and $x :: name$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$
and $C1: \bigwedge Q.\ (\Psi, Q, Q) \in Rel$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P \rightsquigarrow[Rel]\ (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P)$
⟨*proof*⟩

lemma *outputPushResLeft*:
fixes $\Psi :: 'b$
and $x :: name$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes *eqvt Rel*

and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$
and $C1\colon \bigwedge Q.\ (\Psi,\ Q,\ Q) \in Rel$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \rightsquigarrow [Rel]\ M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *outputPushResRight*:
fixes $\Psi :: 'b$
and $x :: name$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a,\ 'b,\ 'c)\ psi$

assumes *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$
and $C1\colon \bigwedge Q.\ (\Psi,\ Q,\ Q) \in Rel$

shows $\Psi \rhd M\langle N\rangle.(\!|\nu x|\!)P \rightsquigarrow [Rel]\ (\!|\nu x|\!)(M\langle N\rangle.P)$
⟨*proof*⟩

lemma *casePushResLeft*:
fixes $\Psi :: 'b$
and $x :: name$
and $Cs :: ('c \times ('a,\ 'b,\ 'c)\ psi)\ list$

assumes *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp map\ fst\ Cs$
and $C1\colon \bigwedge Q.\ (\Psi,\ Q,\ Q) \in Rel$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \rightsquigarrow [Rel]\ Cases\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs)$
⟨*proof*⟩

lemma *casePushResRight*:
fixes $\Psi :: 'b$
and $x :: name$
and $Cs :: ('c \times ('a,\ 'b,\ 'c)\ psi)\ list$

assumes *eqvt Rel*
and $x \sharp \Psi$
and $x \sharp map\ fst\ Cs$
and $C1\colon \bigwedge Q.\ (\Psi,\ Q,\ Q) \in Rel$

shows $\Psi \rhd Cases\ (map\ (\lambda(\varphi,\ P).\ (\varphi,\ (\!|\nu x|\!)P))\ Cs) \rightsquigarrow [Rel]\ (\!|\nu x|\!)(Cases\ Cs)$
⟨*proof*⟩

lemma *resInputCases*[*consumes 5*, *case-names cRes*]:
fixes Ψ :: $'b$
and x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto M(\!|N|\!) \prec P'$
and $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$
and $x \sharp P'$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto M(\!|N|\!) \prec P']\!] \Longrightarrow Prop\ ((\!|\nu x|\!)P')$

shows *Prop* P'
⟨*proof*⟩

lemma *scopeExtLeft*:
fixes x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $x \sharp P$
and $x \sharp \Psi$
and *eqvt Rel*
and *C1*: $\bigwedge \Psi'\ R.\ (\Psi', R, R) \in Rel$
and *C2*: $\bigwedge y\ \Psi'\ R\ S\ zvec.\ [\![y \sharp \Psi'; y \sharp R; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)((\!|\nu* zvec|\!)(R \parallel S)), (\!|\nu* zvec|\!)(R \parallel (\!|\nu y|\!)S)) \in Rel$
and *C3*: $\bigwedge \Psi'\ zvec\ R\ y.\ [\![y \sharp \Psi'; zvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)((\!|\nu* zvec|\!)R), (\!|\nu* zvec|\!)((\!|\nu y|\!)R)) \in Rel$

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

lemma *scopeExtRight*:
fixes x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and Ψ :: $'b$
and Q :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $x \sharp P$
and $x \sharp \Psi$
and *eqvt Rel*

and *C1*: $\bigwedge \Psi'\ R.\ (\Psi,\ R,\ R) \in Rel$
and *C2*: $\bigwedge y\ \Psi'\ R\ S\ zvec.\ [\![y \sharp \Psi';\ y \sharp R;\ zvec \sharp* \Psi]\!] \Longrightarrow (\Psi',\ (\!|\nu * zvec|\!)(R \parallel (\!|\nu y|\!)S),\ (\!|\nu y|\!)((\!|\nu * zvec|\!)(R \parallel S))) \in Rel$

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

lemma *simParComm*:
fixes Ψ :: $'b$
and P :: $('a,\ 'b,\ 'c)\ psi$
and Q :: $('a,\ 'b,\ 'c)\ psi$
and Rel :: $('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *eqvt Rel*
and *C1*: $\bigwedge \Psi'\ R\ S.\ (\Psi',\ R \parallel S,\ S \parallel R) \in Rel$
and *C2*: $\bigwedge \Psi'\ R\ S\ xvec.\ [\![(\Psi',\ R,\ S) \in Rel;\ xvec \sharp* \Psi]\!] \Longrightarrow (\Psi',\ (\!|\nu * xvec|\!)R,\ (\!|\nu * xvec|\!)S) \in Rel$

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

lemma *bangExtLeft*:
fixes Ψ :: $'b$
and P :: $('a,\ 'b,\ 'c)\ psi$

assumes *guarded P*
and $\bigwedge \Psi'\ Q.\ (\Psi',\ Q,\ Q) \in Rel$

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

lemma *bangExtRight*:
fixes Ψ :: $'b$
and P :: $('a,\ 'b,\ 'c)\ psi$

assumes *C1*: $\bigwedge \Psi'\ Q.\ (\Psi',\ Q,\ Q) \in Rel$

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

end

end

theory *Structural-Congruence*
imports *Agent*
begin

inductive *structCong* :: ((*'a*::*fs-name*), (*'b*::*fs-name*), (*'c*::*fs-name*)) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ≡$_s$ -› [*70*, *70*] *70*)
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*: $P \parallel \mathbf{0} \equiv_s P$

| *ResNil*: $(\!|\nu x|\!)\mathbf{0} \equiv_s \mathbf{0}$
| *ResComm*: $(\!|\nu x|\!)((\!|\nu y|\!)P) \equiv_s (\!|\nu y|\!)((\!|\nu x|\!)P)$
| *ScopeExtPar*: $x \sharp P \Longrightarrow (\!|\nu x|\!)(P \parallel Q) \equiv_s P \parallel (\!|\nu x|\!)Q$
| *InputRes*: $[\![x \sharp M; x \sharp xvec; x \sharp N]\!] \Longrightarrow (\!|\nu x|\!)(M(\!|\lambda* xvec\ N|\!).P) \equiv_s M(\!|\lambda* xvec\ N|\!).((\!|\nu x|\!)P)$
| *OutputRes*: $[\![x \sharp M; x \sharp N]\!] \Longrightarrow (\!|\nu x|\!)(M\langle N\rangle.P) \equiv_s M\langle N\rangle.((\!|\nu x|\!)P)$
| *CaseRes*: $x \sharp (map\ fst\ Cs) \Longrightarrow (\!|\nu x|\!)(Cases\ Cs) \equiv_s Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$

| *BangUnfold*: *guarded* $P \Longrightarrow\ !P \equiv_s P \parallel\ !P$

end

theory *Bisim-Struct-Cong*
  imports *Bisim-Pres Sim-Struct-Cong Structural-Congruence*
begin

context *env* **begin**

lemma *bisimParComm*:
  fixes Ψ :: *'b*
  and P :: (*'a*, *'b*, *'c*) *psi*
  and Q :: (*'a*, *'b*, *'c*) *psi*

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

lemma *bisimResComm*:
  fixes x :: *name*
  and Ψ :: *'b*
  and y :: *name*
  and P :: (*'a*, *'b*, *'c*) *psi*

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

lemma *bisimResComm'*:
  fixes x :: *name*

and Ψ :: $'b$
and $xvec$ :: $name\ list$
and P :: $('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P) \sim (\!|\nu{*}xvec|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *bisimScopeExt*:
fixes x :: $name$
and Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$

assumes $x \sharp P$

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

lemma *bisimScopeExtChain*:
fixes $xvec$ :: $name\ list$
and Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* P$

shows $\Psi \rhd (\!|\nu{*}xvec|\!)(P \parallel Q) \sim P \parallel ((\!|\nu{*}xvec|\!)Q)$
⟨*proof*⟩

lemma *bisimParAssoc*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$

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

lemma *bisimParNil*:
fixes P :: $('a, 'b, 'c)\ psi$

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

lemma *bisimResNil*:

fixes $x :: name$
and $\Psi :: 'b$

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

lemma *bisimOutputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp M$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \sim M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *bisimInputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \sim M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *bisimCasePushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $x \sharp (map\ fst\ Cs)$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \sim Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$
⟨*proof*⟩

lemma *bangExt*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes $guarded\ P$

shows $\Psi \rhd !P \sim P \parallel !P$
⟨*proof*⟩

lemma *bisimParPresSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim Q$

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

lemma *bisimScopeExtSym*:
fixes $x :: name$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp Q$

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

lemma *bisimScopeExtChainSym*:
fixes $xvec :: name\ list$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* Q$

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

lemma *bisimParPresAuxSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \otimes \Psi_R \rhd P \sim Q$
and $extractFrame\ R = \langle A_R, \Psi_R \rangle$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$

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

lemma *bangDerivative*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd !P \longmapsto \alpha \prec P'$
and $\Psi \rhd P \sim Q$
and *bn* $\alpha \sharp* \Psi$
and *bn* $\alpha \sharp* P$
and *bn* $\alpha \sharp* Q$
and *bn* $\alpha \sharp*$ *subject* α
and *guarded* Q

obtains Q' R T **where** $\Psi \rhd !Q \longmapsto \alpha \prec Q'$ **and** $\Psi \rhd P' \sim R \parallel !P$ **and** $\Psi \rhd Q' \sim T \parallel !Q$ **and** $\Psi \rhd R \sim T$
and ((*supp* R)::*name set*) ⊆ *supp* P' **and** ((*supp* T)::*name set*) ⊆ *supp* Q'
⟨*proof*⟩

lemma *structCongBisim*:
fixes P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

assumes $P \equiv_s Q$

shows $P \sim Q$
⟨*proof*⟩

lemma *bisimBangPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \sim Q$
and *guarded* P
and *guarded* Q

shows $\Psi \rhd !P \sim !Q$
⟨*proof*⟩

end

end

theory *Weak-Bisimulation*

imports *Weak-Simulation Weak-Stat-Imp Bisim-Struct-Cong*
begin

context *env* **begin**

lemma *monoCoinduct*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \rightsquigarrow <\{(xc,\ xb,\ xa).\ x\ xc\ xb\ xa\}> P) \longrightarrow$
$(\Psi \rhd Q \rightsquigarrow <\{(xb,\ xa,\ xc).\ y\ xb\ xa\ xc\}> P)$
⟨*proof*⟩

lemma *monoCoinduct2*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \lesssim <\{(xc,\ xb,\ xa).\ x\ xc\ xb\ xa\}> P) \longrightarrow$
$(\Psi \rhd Q \lesssim <\{(xb,\ xa,\ xc).\ y\ xb\ xa\ xc\}> P)$
⟨*proof*⟩

coinductive-set *weakBisim* :: $('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
where
step: $[\![\Psi \rhd P \lesssim <weakBisim> Q;\ \Psi \rhd P \rightsquigarrow <weakBisim> Q;$
$\forall \Psi'.\ (\Psi \otimes \Psi',\ P,\ Q) \in weakBisim;\ (\Psi,\ Q,\ P) \in weakBisim]\!] \Longrightarrow (\Psi,\ P,\ Q) \in weakBisim$
monos *monoCoinduct monoCoinduct2*

abbreviation
weakBisimJudge (‹- ⊳ - ≈ -› [70, 70, 70] 65) **where** $\Psi \rhd P \approx Q \equiv (\Psi,\ P,\ Q) \in weakBisim$
abbreviation
weakBisimNilJudge (‹- ≈ -› [70, 70] 65) **where** $P \approx Q \equiv \mathbf{1} \rhd P \approx Q$

lemma *weakBisimCoinductAux*[*consumes 1*]:
fixes $F :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes $(\Psi,\ P,\ Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi \rhd P \lesssim <(X \cup weakBisim)> Q) \wedge$
$(\Psi \rhd P \rightsquigarrow <(X \cup weakBisim)> Q) \wedge$
$(\forall \Psi'.\ (\Psi \otimes \Psi',\ P,\ Q) \in X \vee (\Psi \otimes \Psi',\ P,\ Q) \in weakBisim) \wedge$
$((\Psi,\ Q,\ P) \in X \vee (\Psi,\ Q,\ P) \in weakBisim)$

shows $(\Psi,\ P,\ Q) \in weakBisim$
⟨*proof*⟩

lemma *weakBisimCoinduct*[*consumes 1*, *case-names cStatImp cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \lesssim <(X \cup weakBisim)> S$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow <(X \cup weakBisim)> S$
and $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi', R, S) \in X \Longrightarrow (\Psi' \otimes \Psi'', R, S) \in X \vee \Psi' \otimes \Psi'' \rhd R \approx S$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow (\Psi', S, R) \in X \vee \Psi' \rhd S \approx R$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *weakBisimWeakCoinductAux*[*consumes 1*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lesssim <X> Q \wedge \Psi \rhd P \rightsquigarrow <X> Q \wedge (\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X) \wedge (\Psi, Q, P) \in X$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *weakBisimWeakCoinduct*[*consumes 1*, *case-names cStatImp cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lesssim <X> Q$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow <X> Q$
and $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in weakBisim$
⟨*proof*⟩

lemma *weakBisimE*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \approx Q$

shows $\Psi \rhd P \lessapprox_{<weakBisim>} Q$
and $\Psi \rhd P \rightsquigarrow_{<weakBisim>} Q$
and $\Psi \otimes \Psi' \rhd P \approx Q$
and $\Psi \rhd Q \approx P$
⟨*proof*⟩

lemma *weakBisimI*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$

assumes $\Psi \rhd P \lessapprox_{<weakBisim>} Q$
and $\Psi \rhd P \rightsquigarrow_{<weakBisim>} Q$
and $\forall \Psi'.\ \Psi \otimes \Psi' \rhd P \approx Q$
and $\Psi \rhd Q \approx P$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *weakBisimReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \approx P$
⟨*proof*⟩

lemma *weakBisimClosed*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes $\Psi \rhd P \approx Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \approx (p \cdot Q)$
⟨*proof*⟩

lemma *weakBisimEqvt*[*simp*]:
shows *eqvt weakBisim*
⟨*proof*⟩

lemma *statEqWeakBisim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \approx Q$

and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \approx Q$
⟨*proof*⟩

lemma *weakBisimTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes *PQ*: $\Psi \rhd P \approx Q$
and *QR*: $\Psi \rhd Q \approx R$

shows $\Psi \rhd P \approx R$
⟨*proof*⟩

lemma *strongBisimWeakBisim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *structCongWeakBisim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \equiv_s Q$

shows $P \approx Q$
⟨*proof*⟩

lemma *simTauChain*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $Q' :: ('a, 'b, 'c)\ psi$

assumes $(\Psi, P, Q) \in Rel$
and $\Psi \rhd Q \Longrightarrow\!\!\hat{}_{\tau}\ Q'$
and *Sim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in Rel \Longrightarrow \Psi \rhd P \rightsquigarrow [Rel]\ Q$

obtains P' **where** $\Psi \rhd P \Longrightarrow\!\!\hat{}_{\tau}\ P'$ **and** $(\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *quietBisimNil*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *quiet* P

shows $\Psi \rhd P \approx \mathbf{0}$
⟨*proof*⟩

lemma *weakTransitiveWeakCoinduct*[*case-names cStatImp cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *p*: $(\Psi, P, Q) \in X$
and *Eqvt*: *eqvt* X
and *rStatImp*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox <X> Q$
and *rSim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow <(\{(\Psi, P, Q) \mid \Psi\ P\ Q.\ \exists P'\ Q'.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\})> Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$
and *rSym*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *weakTransitiveCoinduct*[*case-names cStatImp cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *p*: $(\Psi, P, Q) \in X$
and *Eqvt*: *eqvt* X
and *rStatImp*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox <(X \cup weakBisim)> Q$
and *rSim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow <(\{(\Psi, P, Q) \mid \Psi\ P\ Q.\ \exists P'\ Q'.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in (X \cup weakBisim) \wedge \Psi \rhd Q' \sim Q\})> Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X \cup weakBisim$
and *rSym*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X \cup weakBisim$

shows $\Psi \rhd P \approx Q$

⟨*proof*⟩

lemma *weakTransitiveCoinduct2*[*case-names cStatImp cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and X :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *p*: $(\Psi, P, Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatImp*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox <X> Q$
and *rSim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow <(\{(\Psi, P, Q) \mid \Psi\ P\ Q.\ \exists P'\ Q'.\ \Psi \rhd P \approx P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\})> Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$
and *rSym*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

end

end

theory *Weak-Sim-Pres*
imports *Sim-Pres Weak-Simulation Weak-Stat-Imp*
begin

context *env* **begin**

lemma *weakInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes *PRelQ*: $\bigwedge Tvec\ \Psi'.\ length\ xvec = length\ Tvec \Longrightarrow (\Psi \otimes \Psi', P[xvec::=Tvec], Q[xvec::=Tvec]) \in Rel$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \rightsquigarrow <Rel> M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakOutputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$

assumes *PRelQ*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', P, Q) \in Rel$

shows $\Psi \rhd M\langle N\rangle.P \rightsquigarrow <Rel> M\langle N\rangle.Q$
⟨*proof*⟩

lemma *resTauCases*[*consumes 3*, *case-names cRes*]:
fixes Ψ :: $'b$
and x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and P' :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto \tau \prec P'$
and $x \sharp \Psi$
and $x \sharp P'$
and *rScope*: $\bigwedge P'.\ [\![\Psi \rhd P \longmapsto \tau \prec P']\!] \Longrightarrow Prop\ ((\!|\nu x|\!)P')$

shows *Prop* P'
⟨*proof*⟩

lemma *weakResPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and x :: *name*
and *Rel'* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow <Rel> Q$
and *eqvt Rel'*
and $x \sharp \Psi$
and $Rel \subseteq Rel'$
and *C1*: $\bigwedge \Psi'\ R\ S\ y.\ [\![(\Psi', R, S) \in Rel;\ y \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)R, (\!|\nu y|\!)S) \in Rel'$

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

lemma *weakResChainPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow <Rel> Q$
and *eqvt Rel*
and $xvec \sharp* \Psi$
and *C1*: $\bigwedge \Psi' R S yvec. [\![(\Psi', R, S) \in Rel; yvec \sharp* \Psi]\!] \Longrightarrow (\Psi', (\!|\nu * yvec|\!) R, (\!|\nu * yvec|\!) S) \in Rel$

shows $\Psi \rhd (\!|\nu * xvec|\!) P \rightsquigarrow <Rel> (\!|\nu * xvec|\!) Q$
⟨*proof*⟩

lemma *parTauCases*[*consumes 1*, *case-names cPar1 cPar2 cComm1 cComm2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$
and $C :: 'd::fs\text{-}name$

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto \tau \prec R$
and *rPar1*: $\bigwedge P' A_Q \Psi_Q. [\![\Psi \otimes \Psi_Q \rhd P \longmapsto \tau \prec P';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$A_Q \sharp* \Psi;\ A_Q \sharp* P;\ A_Q \sharp* Q;\ A_Q \sharp* C]\!] \Longrightarrow Prop\ (P' \parallel Q)$
and *rPar2*: $\bigwedge Q' A_P \Psi_P. [\![\Psi \otimes \Psi_P \rhd Q \longmapsto \tau \prec Q';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$A_P \sharp* \Psi;\ A_P \sharp* P;\ A_P \sharp* Q;\ A_P \sharp* C]\!] \Longrightarrow Prop\ (P \parallel Q')$
and *rComm1*: $\bigwedge \Psi_Q\ M\ N\ P'\ A_P\ \Psi_P\ K\ xvec\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;$
$distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$
and *rComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ K\ Q'\ A_Q.$
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P';\ extractFrame\ P = \langle A_P, \Psi_P\rangle;\ distinct\ A_P;$
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q';\ extractFrame\ Q = \langle A_Q, \Psi_Q\rangle;\ distinct\ A_Q;$
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K;\ distinct\ xvec;$
$A_P \sharp* \Psi;\ A_P \sharp* \Psi_Q;\ A_P \sharp* P;\ A_P \sharp* M;\ A_P \sharp* N;\ A_P \sharp* P';\ A_P \sharp* Q;\ A_P \sharp* xvec;\ A_P \sharp* Q';\ A_P \sharp* A_Q;\ A_P \sharp* C;$
$A_Q \sharp* \Psi;\ A_Q \sharp* \Psi_P;\ A_Q \sharp* P;\ A_Q \sharp* K;\ A_Q \sharp* N;\ A_Q \sharp* P';\ A_Q \sharp* Q;\ A_Q \sharp* xvec;\ A_Q \sharp* Q';\ A_Q \sharp* C;$
$xvec \sharp* \Psi;\ xvec \sharp* \Psi_P;\ xvec \sharp* P;\ xvec \sharp* M;\ xvec \sharp* K;\ xvec \sharp* Q;\ xvec \sharp* \Psi_Q;\ xvec \sharp* C]\!] \Longrightarrow$
$Prop\ ((\!|\nu * xvec|\!)(P' \parallel Q'))$

shows *Prop R*

⟨*proof*⟩

lemma *weakParPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PRelQ*: $\bigwedge A_R\ \Psi_R.\ [\![extractFrame\ R = \langle A_R, \Psi_R\rangle;\ A_R \sharp* \Psi;\ A_R \sharp* P;\ A_R \sharp* Q]\!] \Longrightarrow (\Psi \otimes \Psi_R, P, Q) \in Rel$
and *Eqvt*: $eqvt\ Rel$
and *Eqvt'*: $eqvt\ Rel'$

and *Sim*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow \Psi' \rhd S \rightsquigarrow <Rel> T$
and *Sym*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow (\Psi', T, S) \in Rel$
and *Ext*: $\bigwedge \Psi'\ S\ T\ \Psi'.\ [\![(\Psi', S, T) \in Rel]\!] \Longrightarrow (\Psi' \otimes \Psi'', S, T) \in Rel$
and *StatImp*: $\bigwedge \Psi'\ S\ T.\ (\Psi', S, T) \in Rel \Longrightarrow \Psi' \rhd S \lesssim <Rel> T$
and *C1*: $\bigwedge \Psi'\ S\ T\ A_U\ \Psi_U\ U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel;\ extractFrame\ U = \langle A_U, \Psi_U\rangle;\ A_U \sharp* \Psi';\ A_U \sharp* S;\ A_U \sharp* T]\!] \Longrightarrow (\Psi', S \parallel U, T \parallel U) \in Rel'$
and *C2*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel';\ xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!) S, (\!|\nu * xvec|\!)\ T) \in Rel'$
and *C3*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel;\ \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', S, T) \in Rel$

shows $\Psi \rhd P \parallel R \rightsquigarrow <Rel'> Q \parallel R$
⟨*proof*⟩
unbundle *no relcomp-syntax*
lemma *weakSimBangPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Rel'' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in Rel$
and $eqvt\ Rel''$
and $guarded\ P$
and $guarded\ Q$
and $Rel' Rel$: $Rel' \subseteq Rel$

and *FrameParPres*: $\bigwedge \Psi'\ \Psi_U\ S\ T\ U\ A_U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel;\ extractFrame\ U = \langle A_U, \Psi_U\rangle;\ A_U \sharp* \Psi';\ A_U \sharp* S;\ A_U \sharp* T]\!] \Longrightarrow (\Psi', U \parallel S, U \parallel T) \in Rel$
and *C1*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi', S, T) \in Rel;\ guarded\ S;\ guarded\ T]\!] \Longrightarrow (\Psi', U \parallel\ !S, U \parallel\ !T) \in Rel''$
and *ResPres*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel;\ xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!) S, (\!|\nu * xvec|\!)\ T) \in Rel$

and *ResPres'*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi',\ S,\ T) \in Rel';\ xvec\ \sharp*\ \Psi']\!] \Longrightarrow (\Psi',\ (\!|\nu*xvec|\!)S,\ (\!|\nu*xvec|\!)\,T) \in Rel'$

and *Closed*: $\bigwedge \Psi'\ S\ T\ p.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow ((p::name\ prm) \cdot \Psi',\ p \cdot S,\ p \cdot T) \in Rel$
and *Closed'*: $\bigwedge \Psi'\ S\ T\ p.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow ((p::name\ prm) \cdot \Psi',\ p \cdot S,\ p \cdot T) \in Rel'$
and *StatEq*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi',\ S,\ T) \in Rel;\ \Psi' \simeq \Psi']\!] \Longrightarrow (\Psi'',\ S,\ T) \in Rel$
and *StatEq'*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi',\ S,\ T) \in Rel';\ \Psi' \simeq \Psi']\!] \Longrightarrow (\Psi'',\ S,\ T) \in Rel'$
and *Trans*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi',\ S,\ T) \in Rel;\ (\Psi',\ T,\ U) \in Rel]\!] \Longrightarrow (\Psi',\ S,\ U) \in Rel$
and *Trans'*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi',\ S,\ T) \in Rel';\ (\Psi',\ T,\ U) \in Rel']\!] \Longrightarrow (\Psi',\ S,\ U) \in Rel'$

and *cSim*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow \Psi' \rhd S \rightsquigarrow <Rel>\ T$
and *cExt*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi' \otimes \Psi'',\ S,\ T) \in Rel$
and *cExt'*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi' \otimes \Psi'',\ S,\ T) \in Rel'$
and *cSym*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi',\ T,\ S) \in Rel$
and *cSym'*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi',\ T,\ S) \in Rel'$

and *ParPres*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi',\ S \parallel U,\ T \parallel U) \in Rel$
and *ParPres2*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi',\ S \parallel S,\ T \parallel T) \in Rel$
and *ParPres'*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi',\ U \parallel S,\ U \parallel T) \in Rel'$

and *Assoc*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S \parallel (T \parallel U),\ (S \parallel T) \parallel U) \in Rel$
and *Assoc'*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S \parallel (T \parallel U),\ (S \parallel T) \parallel U) \in Rel'$
and *ScopeExt*: $\bigwedge xvec\ \Psi'\ T\ S.\ [\![xvec\ \sharp*\ \Psi';\ xvec\ \sharp*\ T]\!] \Longrightarrow (\Psi',\ (\!|\nu*xvec|\!)(S \parallel T),\ ((\!|\nu*xvec|\!)S) \parallel T) \in Rel$
and *ScopeExt'*: $\bigwedge xvec\ \Psi'\ T\ S.\ [\![xvec\ \sharp*\ \Psi';\ xvec\ \sharp*\ T]\!] \Longrightarrow (\Psi',\ (\!|\nu*xvec|\!)(S \parallel T),\ ((\!|\nu*xvec|\!)S) \parallel T) \in Rel'$

and *Compose*: $\bigwedge \Psi'\ S\ T\ U\ O.\ [\![(\Psi',\ S,\ T) \in Rel;\ (\Psi',\ T,\ U) \in Rel'';\ (\Psi',\ U,\ O) \in Rel']\!] \Longrightarrow (\Psi',\ S,\ O) \in Rel''$

and *rBangActE*: $\bigwedge \Psi'\ S\ \alpha\ S'.\ [\![\Psi' \rhd\ !S \longmapsto \alpha \prec S';\ guarded\ S;\ bn\ \alpha\ \sharp*\ S;\ \alpha \neq \tau;\ bn\ \alpha\ \sharp*\ subject\ \alpha]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd S \longmapsto \alpha \prec T \wedge (\mathbf{1},\ S',\ T \parallel\ !S) \in Rel'$
and *rBangTauE*: $\bigwedge \Psi'\ S\ S'.\ [\![\Psi' \rhd\ !S \longmapsto \tau \prec S';\ guarded\ S]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd S \parallel S \longmapsto \tau \prec T \wedge (\mathbf{1},\ S',\ T \parallel\ !S) \in Rel'$
and *rBangTauI*: $\bigwedge \Psi'\ S\ S'.\ [\![\Psi' \rhd S \parallel S \Longrightarrow\hat{}_\tau\ S';\ guarded\ S]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd\ !S \Longrightarrow\hat{}_\tau\ T \wedge (\Psi',\ T,\ S' \parallel\ !S) \in Rel'$
shows $\Psi \rhd R \parallel\ !P \rightsquigarrow <Rel''>\ R \parallel\ !Q$
⟨*proof*⟩
unbundle *relcomp-syntax*
end

end

theory *Weak-Stat-Imp-Pres*
imports *Weak-Stat-Imp*
begin

context *env* **begin**

lemma *weakStatImpInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes *PRelQ*: $\bigwedge\Psi'.\ (\Psi \otimes \Psi', M(\!|\lambda * xvec\ N|\!).P, M(\!|\lambda * xvec\ N|\!).Q) \in Rel$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \lessapprox <Rel> M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakStatImpOutputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and N :: $'a$

assumes *PRelQ*: $\bigwedge\Psi'.\ (\Psi \otimes \Psi', M\langle N\rangle.P, M\langle N\rangle.Q) \in Rel$

shows $\Psi \rhd M\langle N\rangle.P \lessapprox <Rel> M\langle N\rangle.Q$
⟨*proof*⟩

lemma *weakStatImpResPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*
and Q :: $('a, 'b, 'c)$ *psi*
and x :: *name*
and *Rel'* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*$)$ *set*

assumes *PSimQ*: $\Psi \rhd P \lessapprox <Rel> Q$
and *eqvt Rel*
and $x \sharp \Psi$
and *C1*: $\bigwedge\Psi'\ R\ S\ y.\ [\![(\Psi', R, S) \in Rel;\ y \sharp \Psi']\!] \Longrightarrow (\Psi', (\!|\nu y|\!)R, (\!|\nu y|\!)S) \in Rel'$

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

lemma *weakStatImpParPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Rel :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)\ psi$
and R :: $('a, 'b, 'c)\ psi$
and Rel' :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PStatImpQ*: $\bigwedge A_R\ \Psi_R.\ [\![extractFrame\ R = \langle A_R, \Psi_R\rangle;\ A_R \sharp* \Psi;\ A_R \sharp* P;\ A_R \sharp* Q]\!] \Longrightarrow \Psi \otimes \Psi_R \rhd P \lessapprox <Rel> Q$
and $xvec \sharp* \Psi$
and *Eqvt*: *eqvt Rel*

and *C1*: $\bigwedge \Psi'\ S\ T\ A_U\ \Psi_U\ U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel;\ extractFrame\ U = \langle A_U, \Psi_U\rangle;\ A_U \sharp* \Psi';\ A_U \sharp* S;\ A_U \sharp* T]\!] \Longrightarrow (\Psi', S \parallel U, T \parallel U) \in Rel'$
and *C2*: $\bigwedge \Psi'\ S\ T\ yvec.\ [\![(\Psi', S, T) \in Rel';\ yvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu* yvec|\!) S, (\!|\nu* yvec|\!)\ T) \in Rel'$
and *C3*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel;\ \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', S, T) \in Rel$

shows $\Psi \rhd (\!|\nu* xvec|\!)(P \parallel R) \lessapprox <Rel'> (\!|\nu* xvec|\!)(Q \parallel R)$
⟨*proof*⟩

end

end

theory *Weak-Bisim-Pres*
imports *Weak-Bisimulation Weak-Sim-Pres Weak-Stat-Imp-Pres*
begin

context *env* **begin**

lemma *weakBisimInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$
and M :: $'a$
and $xvec$ :: *name list*
and N :: $'a$

assumes $\bigwedge Tvec.\ length\ xvec = length\ Tvec \Longrightarrow \Psi \rhd P[xvec ::= Tvec] \approx Q[xvec ::= Tvec]$

shows $\Psi \rhd M(\!|\lambda* xvec\ N|\!).P \approx M(\!|\lambda* xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakBisimOutputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $\Psi \rhd P \approx Q$

shows $\Psi \rhd M\langle N\rangle.P \approx M\langle N\rangle.Q$
⟨*proof*⟩

lemma *weakBisimResPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and x :: *name*

assumes $\Psi \rhd P \approx Q$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!)P \approx (\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *weakBisimResChainPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and *xvec* :: *name list*

assumes $\Psi \rhd P \approx Q$
and *xvec* $\sharp* \Psi$

shows $\Psi \rhd (\!|\nu* xvec|\!)P \approx (\!|\nu* xvec|\!)Q$
⟨*proof*⟩

lemma *weakBisimParPresAux*:
fixes Ψ :: $'b$
and Ψ_R :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*
and A_R :: *name list*

assumes $\Psi \otimes \Psi_R \rhd P \approx Q$
and *FrR*: *extractFrame* $R = \langle A_R, \Psi_R\rangle$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$
and $A_R \sharp* Q$

shows $\Psi \rhd P \parallel R \approx Q \parallel R$

⟨*proof*⟩

lemma *weakBisimParPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \approx Q$

shows $\Psi \rhd P \parallel R \approx Q \parallel R$
⟨*proof*⟩

end

end

theory *Weak-Bisim-Struct-Cong*
imports *Weak-Bisim-Pres Bisim-Struct-Cong*
begin

context *env* **begin**

lemma *weakBisimParComm*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

shows $\Psi \rhd P \parallel Q \approx Q \parallel P$
⟨*proof*⟩

lemma *weakBisimResComm*:
fixes x :: *name*
and Ψ :: $'b$
and y :: *name*
and P :: $('a, 'b, 'c)$ *psi*

assumes $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \approx (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *weakBisimResComm'*:
fixes x :: *name*
and Ψ :: $'b$
and *xvec* :: *name list*
and P :: $('a, 'b, 'c)$ *psi*

assumes $x \sharp \Psi$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu{*}xvec|\!)P) \approx (\!|\nu{*}xvec|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *weakBisimScopeExt*:
fixes $x :: name$
and $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp P$

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

lemma *weakBisimScopeExtChain*:
fixes $xvec :: name\ list$
and $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* P$

shows $\Psi \rhd (\!|\nu{*}xvec|\!)(P \parallel Q) \approx P \parallel ((\!|\nu{*}xvec|\!)Q)$
⟨*proof*⟩

lemma *weakBisimParAssoc*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \approx P \parallel (Q \parallel R)$
⟨*proof*⟩

lemma *weakBisimParNil*:
fixes $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel \mathbf{0} \approx P$
⟨*proof*⟩

lemma *weakBisimResNil*:
fixes $x :: name$
and $\Psi :: 'b$

assumes $x \sharp \Psi$

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

lemma *weakBisimOutputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \approx M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakBisimInputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \approx M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakBisimCasePushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $x \sharp \Psi$
and $x \sharp (map\ fst\ Cs)$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \approx Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$
⟨*proof*⟩

lemma *weakBangExt*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *guarded P*

shows $\Psi \rhd !P \approx P \parallel !P$
⟨*proof*⟩

lemma *weakBisimParSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx Q$

shows $\Psi \rhd R \parallel P \approx R \parallel Q$
⟨*proof*⟩

lemma *weakBisimScopeExtSym*:
fixes $x :: name$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp Q$

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

lemma *weakBisimScopeExtChainSym*:
fixes $xvec :: name\ list$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* Q$

shows $\Psi \rhd (\!|\nu * xvec|\!)(P \parallel Q) \approx ((\!|\nu * xvec|\!)P) \parallel Q$
⟨*proof*⟩

lemma *weakBisimParPresAuxSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \otimes \Psi_R \rhd P \approx Q$
and $extractFrame\ R = \langle A_R, \Psi_R \rangle$
and $A_R \sharp* \Psi$
and $A_R \sharp* P$

and $A_R \sharp* Q$

shows $\Psi \rhd R \parallel P \approx R \parallel Q$
⟨*proof*⟩

lemma *weakBisimParPresSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx Q$

shows $\Psi \rhd R \parallel P \approx R \parallel Q$
⟨*proof*⟩

lemma *guardedFrameStatEq*:
fixes $P :: ('a, 'b, 'c)\ psi$

assumes *guarded P*

shows *extractFrame P* $\simeq_F \langle \varepsilon, \mathbf{1} \rangle$
⟨*proof*⟩

lemma *guardedInsertAssertion*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$

assumes *guarded P*

shows *insertAssertion* (*extractFrame P*) $\Psi \simeq_F \langle \varepsilon, \Psi \rangle$
⟨*proof*⟩

lemma *insertDoubleAssertionStatEq'*:
fixes $F :: 'b\ frame$
and $\Psi :: 'b$
and $\Psi' :: 'b$

shows *insertAssertion*(*insertAssertion F* Ψ) $\Psi' \simeq_F$ (*insertAssertion F*) ($\Psi' \otimes \Psi$)
⟨*proof*⟩

lemma *bangActE*:
assumes $\Psi \rhd !P \longmapsto \alpha \prec P'$
and *bn* $\alpha \sharp*$ *subject* α
and *guarded P*
and $\alpha \neq \tau$
and *bn* $\alpha \sharp* P$

obtains Q **where** $\Psi \rhd P \longmapsto \alpha \prec Q$ **and** $P' \sim Q \parallel !P$
⟨*proof*⟩

lemma *bangTauE*:
assumes $\Psi \rhd !P \longmapsto \tau \prec P'$
and *guarded* P

obtains Q **where** $\Psi \rhd P \parallel P \longmapsto \tau \prec Q$ **and** $P' \sim Q \parallel !P$
⟨*proof*⟩

lemma *tauBangI*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \parallel P \longmapsto \tau \prec P'$
and *guarded* P

obtains Q **where** $\Psi \rhd !P \longmapsto \tau \prec Q$ **and** $Q \sim P' \parallel !P$
⟨*proof*⟩

lemma *tauChainBangI*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \parallel P \Longrightarrow\!\hat{}_{\tau}\ P'$
and *guarded* P

obtains Q **where** $\Psi \rhd !P \Longrightarrow\!\hat{}_{\tau}\ Q$ **and** $\Psi \rhd Q \sim P' \parallel !P$
⟨*proof*⟩

lemma *weakBisimBangPresAux*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx Q$
and *guarded* P
and *guarded* Q

shows $\Psi \rhd R \parallel !P \approx R \parallel !Q$
⟨*proof*⟩

lemma *weakBisimBangPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx Q$
and *guarded P*
and *guarded Q*

shows $\Psi \rhd !P \approx !Q$
⟨*proof*⟩

end

end

theory *Close-Subst*
imports *Agent*
begin

context *substPsi*
begin

definition *closeSubst* :: ('b::*fs-name* × ('a::*fs-name*, 'b, 'c::*fs-name*) *psi* × ('a, 'b, 'c) *psi*) *set* ⇒ ('b × ('a, 'b, 'c) *psi* × ('a, 'b, 'c) *psi*) *set*
where *closeSubst Rel* ≡ {(Ψ, *P*, *Q*) | Ψ *P Q*. (∀σ. *wellFormedSubst* σ ⟶ (Ψ, *P*[<σ>], *Q*[<σ>]) ∈ *Rel*)}

lemma *closeSubstI*:
fixes Ψ :: 'b
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes ⋀σ. *wellFormedSubst* σ ⟹ (Ψ, *P*[<σ>], *Q*[<σ>]) ∈ *Rel*

shows (Ψ, *P*, *Q*) ∈ *closeSubst Rel*
⟨*proof*⟩

lemma *closeSubstE*:
fixes Ψ :: 'b
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assumes (Ψ, *P*, *Q*) ∈ *closeSubst Rel*
and *wellFormedSubst* σ

shows (Ψ, *P*[<σ>], *Q*[<σ>]) ∈ *Rel*
⟨*proof*⟩

lemma *closeSubstClosed*:
fixes Ψ :: 'b

and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes *eqvt Rel*
and $(\Psi, P, Q) \in closeSubst\ Rel$

shows $(p \cdot \Psi, p \cdot P, p \cdot Q) \in closeSubst\ Rel$
⟨*proof*⟩

lemma *closeSubstEqvt*:
assumes *eqvt Rel*

shows *eqvt*(*closeSubst Rel*)
⟨*proof*⟩

lemma *closeSubstUnfold*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assumes $(\Psi, P, Q) \in closeSubst\ Rel$
and *wellFormedSubst* σ

shows $(\Psi, P[<\sigma>], Q[<\sigma>]) \in closeSubst\ Rel$
⟨*proof*⟩

end

end

theory *Bisim-Subst*
imports *Bisim-Struct-Cong Close-Subst*
begin

context *env* **begin**

abbreviation
bisimSubstJudge (‹- ▷ - ∼$_s$ -› [70, 70, 70] 65) **where** $\Psi \rhd P \sim_s Q \equiv (\Psi, P, Q) \in closeSubst\ bisim$
abbreviation
bisimSubstNilJudge (‹- ∼$_s$ -› [70, 70] 65) **where** $P \sim_s Q \equiv SBottom' \rhd P \sim_s Q$

lemmas *bisimSubstClosed*[*eqvt*] = *closeSubstClosed*[*OF bisimEqvt*]
lemmas *bisimSubstEqvt*[*simp*] = *closeSubstEqvt*[*OF bisimEqvt*]

lemma *bisimSubstOutputPres*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes $\Psi \rhd P \sim_s Q$

shows $\Psi \rhd M\langle N\rangle.P \sim_s M\langle N\rangle.Q$
⟨*proof*⟩

lemma *seqSubstInputChain*[*simp*]:
fixes *xvec* :: *name list*
and N :: $'a$
and P :: $('a, 'b, 'c)\ psi$
and σ :: $(name\ list \times 'a\ list)\ list$

assumes $xvec \sharp* \sigma$

shows $seqSubs'\ (inputChain\ xvec\ N\ P)\ \sigma = inputChain\ xvec\ (substTerm.seqSubst\ N\ \sigma)\ (seqSubs\ P\ \sigma)$
⟨*proof*⟩

lemma *bisimSubstInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$
and Q :: $('a, 'b, 'c)\ psi$
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $\Psi \rhd P \sim_s Q$
and $xvec \sharp* \Psi$
and *distinct xvec*

shows $\Psi \rhd M(\!|\lambda* xvec\ N|\!).P \sim_s M(\!|\lambda* xvec\ N|\!).Q$
⟨*proof*⟩

lemma *bisimSubstCasePresAux*:
fixes Ψ :: $'b$
and *CsP* :: $('c \times ('a, 'b, 'c)\ psi)\ list$
and *CsQ* :: $('c \times ('a, 'b, 'c)\ psi)\ list$

assumes *C1*: $\bigwedge\varphi\ P.\ (\varphi, P)\ mem\ CsP \Longrightarrow \exists\, Q.\ (\varphi, Q)\ mem\ CsQ \wedge guarded\ Q \wedge \Psi \rhd P \sim_s Q$
and *C2*: $\bigwedge\varphi\ Q.\ (\varphi, Q)\ mem\ CsQ \Longrightarrow \exists\, P.\ (\varphi, P)\ mem\ CsP \wedge guarded\ P \wedge \Psi \rhd P \sim_s Q$

shows $\Psi \rhd Cases\ CsP \sim_s Cases\ CsQ$
⟨*proof*⟩

lemma *bisimSubstReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \sim_s P$
⟨*proof*⟩

lemma *bisimSubstTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim_s Q$
and $\Psi \rhd Q \sim_s R$

shows $\Psi \rhd P \sim_s R$
⟨*proof*⟩

lemma *bisimSubstSymmetric*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim_s Q$

shows $\Psi \rhd Q \sim_s P$
⟨*proof*⟩

lemma *bisimSubstParPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim_s Q$

shows $\Psi \rhd P \parallel R \sim_s Q \parallel R$
⟨*proof*⟩

lemma *bisimSubstResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $x :: name$

assumes $\Psi \rhd P \sim_s Q$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!) P \sim_s (\!|\nu x|\!) Q$
⟨*proof*⟩

lemma *bisimSubstBangPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim_s Q$
and *guarded P*
and *guarded Q*

shows $\Psi \rhd !P \sim_s !Q$
⟨*proof*⟩

lemma *substNil*[*simp*]:
fixes *xvec* :: *name list*
and *Tvec* :: *'a list*

assumes *wellFormedSubst* σ
and *distinct xvec*

shows $(\mathbf{0}[<\sigma>]) = \mathbf{0}$
⟨*proof*⟩

lemma *bisimSubstParNil*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel \mathbf{0} \sim_s P$
⟨*proof*⟩

lemma *bisimSubstParComm*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel Q \sim_s Q \parallel P$
⟨*proof*⟩

lemma *bisimSubstParAssoc*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \sim_s P \parallel (Q \parallel R)$
⟨*proof*⟩

lemma *bisimSubstResNil*:
fixes $\Psi :: 'b$
and $x :: name$

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

lemma *seqSubst2*:
fixes $x :: name$
and $P :: ('a, 'b, 'c)\ psi$

assumes *wellFormedSubst* σ
and $x \sharp \sigma$
and $x \sharp P$

shows $x \sharp P[<\sigma>]$
⟨*proof*⟩

notation *substTerm.seqSubst* ($\langle$-[<->]$\rangle$ [100, 100] 100)

lemma *bisimSubstScopeExt*:
fixes $\Psi :: 'b$
and $x :: name$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $x \sharp P$

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

lemma *bisimSubstCasePushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $Cs :: ('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $x \sharp map\ fst\ Cs$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \sim_s Cases\ map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs$
⟨*proof*⟩

lemma *bisimSubstOutputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $N :: 'a$

and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp M$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \sim_s M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *bisimSubstInputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$
and $xvec :: name\ list$
and $N :: 'a$

assumes $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \sim_s M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *bisimSubstResComm*:
fixes $x :: name$
and $y :: name$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \sim_s (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *bisimSubstExtBang*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *guarded P*

shows $\Psi \rhd !P \sim_s P \parallel !P$
⟨*proof*⟩

lemma *structCongBisimSubst*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \equiv_s Q$

shows $P \sim_s Q$
⟨*proof*⟩

end

end

theory *Weak-Bisim-Subst*
imports *Weak-Bisim-Struct-Cong Weak-Bisim-Pres Bisim-Subst*
begin

context *env* **begin**

abbreviation
weakBisimSubstJudge (‹- ▷ - ≈$_s$ -› [*70*, *70*, *70*] *65*) **where** Ψ ▷ *P* ≈$_s$ *Q* ≡ (Ψ, *P*, *Q*) ∈ *closeSubst weakBisim*
abbreviation
weakBisimSubstNilJudge (‹- ≈$_s$ -› [*70*, *70*] *65*) **where** *P* ≈$_s$ *Q* ≡ **1** ▷ *P* ≈$_s$ *Q*

lemmas *weakBisimSubstClosed*[*eqvt*] = *closeSubstClosed*[*OF weakBisimEqvt*]
lemmas *weakBisimEqvt*[*simp*] = *closeSubstEqvt*[*OF weakBisimEqvt*]

lemma *strongBisimSubstWeakBisimSubst*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Q* :: (*'a*, *'b*, *'c*) *psi*

assumes Ψ ▷ *P* ∼$_s$ *Q*

shows Ψ ▷ *P* ≈$_s$ *Q*
⟨*proof*⟩

lemma *weakBisimSubstOutputPres*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Q* :: (*'a*, *'b*, *'c*) *psi*
and *M* :: *'a*
and *N* :: *'a*

assumes Ψ ▷ *P* ≈$_s$ *Q*

shows Ψ ▷ *M*⟨*N*⟩.*P* ≈$_s$ *M*⟨*N*⟩.*Q*
⟨*proof*⟩

lemma *bisimSubstInputPres*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *Q* :: (*'a*, *'b*, *'c*) *psi*
and *M* :: *'a*
and *xvec* :: *name list*
and *N* :: *'a*

assumes Ψ ▷ *P* ≈$_s$ *Q*
and *xvec* ♯* Ψ

and *distinct xvec*

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \approx_s M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakBisimSubstReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \approx_s P$
⟨*proof*⟩

lemma *bisimSubstTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx_s Q$
and $\Psi \rhd Q \approx_s R$

shows $\Psi \rhd P \approx_s R$
⟨*proof*⟩

lemma *weakBisimSubstSymmetric*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx_s Q$

shows $\Psi \rhd Q \approx_s P$
⟨*proof*⟩

lemma *weakBisimSubstParPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \approx_s Q$

shows $\Psi \rhd P \parallel R \approx_s Q \parallel R$
⟨*proof*⟩

lemma *weakBisimSubstResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

and $x :: name$

assumes $\Psi \rhd P \approx_s Q$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!) P \approx_s (\!|\nu x|\!) Q$
⟨*proof*⟩

lemma *weakBisimSubstParNil*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel \mathbf{0} \approx_s P$
⟨*proof*⟩

lemma *weakBisimSubstParComm*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel Q \approx_s Q \parallel P$
⟨*proof*⟩

lemma *weakBisimSubstParAssoc*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \approx_s P \parallel (Q \parallel R)$
⟨*proof*⟩

lemma *weakBisimSubstResNil*:
fixes $\Psi :: 'b$
and $x :: name$

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

lemma *weakBisimSubstScopeExt*:
fixes $\Psi :: 'b$
and $x :: name$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $x \sharp P$

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

lemma *weakBisimSubstCasePushRes*:
fixes x :: *name*
and Ψ :: $'b$
and Cs :: $('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $x \sharp map\ fst\ Cs$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \approx_s Cases\ map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs$
⟨*proof*⟩

lemma *weakBisimSubstOutputPushRes*:
fixes x :: *name*
and Ψ :: $'b$
and M :: $'a$
and N :: $'a$
and P :: $('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \approx_s M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakBisimSubstInputPushRes*:
fixes x :: *name*
and Ψ :: $'b$
and M :: $'a$
and $xvec$ :: *name list*
and N :: $'a$

assumes $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \approx_s M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakBisimSubstResComm*:
fixes x :: *name*
and y :: *name*

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \approx_s (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *weakBisimSubstExtBang*:

fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *guarded* P

shows $\Psi \rhd !P \approx_s P \parallel !P$
⟨*proof*⟩

end

end

theory *Weakening*
imports *Weak-Bisimulation*
begin

locale *weak* = *env* +
assumes *weaken*: $\Psi \hookrightarrow \Psi \otimes \Psi'$
begin

lemma *entWeaken*:
fixes $\Psi :: 'b$
and $\varphi :: 'c$

assumes $\Psi \vdash \varphi$

shows $\Psi \otimes \Psi' \vdash \varphi$
⟨*proof*⟩

lemma *assertWeaken*:
fixes $\Psi :: 'b$
and $\Psi' :: 'b$

shows $\Psi \hookrightarrow \Psi \otimes \Psi'$
⟨*proof*⟩

lemma *frameWeaken*:
fixes $F :: 'b\ frame$
and $G :: 'b\ frame$

shows $F \hookrightarrow_F F \otimes_F G$
⟨*proof*⟩

lemma *unitAssertWeaken*:
fixes $\Psi :: 'b$

shows $\mathbf{1} \hookrightarrow \Psi$
⟨*proof*⟩

lemma *unitFrameWeaken*:
fixes $F :: 'b\ frame$

shows $\langle \varepsilon, \mathbf{1} \rangle \hookrightarrow_F F$
⟨*proof*⟩

lemma *insertAssertionWeaken*:
fixes $F :: 'b\ frame$
and $\Psi :: 'b$

shows $\langle \varepsilon, \Psi \rangle \hookrightarrow_F insertAssertion\ F\ \Psi$
⟨*proof*⟩

lemma *frameImpStatEq*:
fixes $A_F :: name\ list$
and $\Psi :: 'b$
and $\Psi' :: 'b$
and $\varphi :: 'c$

assumes $(\langle A_F, \Psi \rangle) \vdash_F \varphi$
and $\Psi \simeq \Psi'$

shows $(\langle A_F, \Psi' \rangle) \vdash_F \varphi$
⟨*proof*⟩

lemma *statImpTauDerivative*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \longmapsto \tau \prec P'$

shows $insertAssertion\ (extractFrame\ P)\ \Psi \hookrightarrow_F insertAssertion\ (extractFrame\ P')\ \Psi$
⟨*proof*⟩

lemma *weakenTransition*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rs :: ('a, 'b, 'c)\ residual$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \longmapsto Rs$

shows $\Psi \otimes \Psi' \rhd P \longmapsto Rs$
⟨*proof*⟩

end

end

theory *Weaken-Transition*
imports *Weakening*
begin

context *weak*
begin

definition *weakenTransition* :: *'b* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *'a action* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ *bool* (‹- ▷ - ⟹- ≺ -› [*80*, *80*, *80*, *80*] *80*)
where
$\Psi \rhd P \Longrightarrow \alpha \prec P' \equiv (\exists P''' P''. \Psi \rhd P \Longrightarrow\hat{}_\tau P''' \wedge \Psi \rhd P''' \longmapsto \alpha \prec P'' \wedge \Psi \rhd P'' \Longrightarrow\hat{}_\tau P') \vee (P = P' \wedge \alpha = \tau)$

lemma *weakenTransitionCases*[*consumes 1*, *case-names cBase cStep*]:
assumes $\Psi \rhd P \Longrightarrow \alpha \prec P'$
and *Prop* (τ) *P*
and $\bigwedge P''' P''. [\![\Psi \rhd P \Longrightarrow\hat{}_\tau P'''; \Psi \rhd P''' \longmapsto \alpha \prec P''; \Psi \rhd P'' \Longrightarrow\hat{}_\tau P']\!] \Longrightarrow$ *Prop* α *P'*

shows *Prop* α *P'*
⟨*proof*⟩

lemma *statImpTauChainDerivative*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *P'* :: (*'a*, *'b*, *'c*) *psi*

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau P'$

shows *insertAssertion* (*extractFrame P*) $\Psi \hookrightarrow_F$ *insertAssertion* (*extractFrame P'*) Ψ
⟨*proof*⟩

lemma *weakenTauChain*:
fixes Ψ :: *'b*
and *P* :: (*'a*, *'b*, *'c*) *psi*
and *P'* :: (*'a*, *'b*, *'c*) *psi*
and Ψ' :: *'b*

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau P'$
shows $\Psi \otimes \Psi' \rhd P \Longrightarrow\hat{}_\tau P'$
⟨*proof*⟩

end

end

theory *Weaken-Stat-Imp*
imports *Weaken-Transition*
begin

context *weak* **begin**

definition
weakenStatImp :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow$
$('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set \Rightarrow$
$('a, 'b, 'c)\ psi \Rightarrow bool\ (\langle\text{-} \rhd \text{-} \lessapprox_w \text{<->} \text{-}\rangle\ [80, 80, 80, 80]\ 80)$
where $\Psi \rhd P \lessapprox_w <Rel> Q \equiv \exists Q'.\ \Psi \rhd Q \Longrightarrow\!\!\hat{}_\tau\ Q' \wedge insertAssertion(extractFrame\ P)\ \Psi \hookrightarrow_F insertAssertion(extractFrame\ Q')\ \Psi \wedge (\Psi, P, Q') \in Rel$

lemma *weakenStatImpMonotonic*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $A :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $B :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $\Psi \rhd P \lessapprox_w <A> Q$
and $A \subseteq B$

shows $\Psi \rhd P \lessapprox_w <B> Q$
⟨*proof*⟩

lemma *weakenStatImpI*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes $\Psi \rhd Q \Longrightarrow\!\!\hat{}_\tau\ Q'$
and $insertAssertion(extractFrame\ P)\ \Psi \hookrightarrow_F insertAssertion(extractFrame\ Q')\ \Psi$
and $(\Psi, P, Q') \in Rel$

shows $\Psi \rhd P \lessapprox_w <Rel> Q$
⟨*proof*⟩

lemma *weakenStatImpE*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \lessapprox_w <Rel> Q$

obtains Q' **where** $\Psi \rhd Q \Longrightarrow\hat{}_{\tau}\ Q'$ **and** *insertAssertion*(*extractFrame* P) Ψ $\hookrightarrow_F$ *insertAssertion*(*extractFrame* Q') Ψ **and** $(\Psi, P, Q') \in Rel$
⟨*proof*⟩

lemma *weakStatImpWeakenStatImp*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)$ *psi*

assumes *cSim*: $\Psi \rhd P \lessapprox <Rel> Q$
and *cStatEq*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ [\![(\Psi', R, S) \in Rel;\ \Psi' \simeq \Psi']\!] \Longrightarrow (\Psi'', R, S) \in Rel$

shows $\Psi \rhd P \lessapprox_w <Rel> Q$
⟨*proof*⟩

lemma *weakenStatImpWeakStatImp*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and Q :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \lessapprox_w <Rel> Q$
and *cExt*: $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi', R, S) \in Rel \Longrightarrow (\Psi' \otimes \Psi'', R, S) \in Rel$

shows $\Psi \rhd P \lessapprox <Rel> Q$
⟨*proof*⟩

end

end

theory *Weaken-Simulation*
imports *Weaken-Stat-Imp*
begin

context *weak*
begin

definition
weakenSimulation :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow$
$('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set \Rightarrow$
$('a, 'b, 'c)\ psi \Rightarrow bool$ (‹- ⊳ - ⇝$_w$<-> -› [80, 80, 80, 80] 80)
where
$\Psi \rhd P \rightsquigarrow_w <Rel> Q \equiv \forall \alpha\ Q'.\ \Psi \rhd Q \longmapsto \alpha \prec Q' \longrightarrow bn\ \alpha\ \sharp*\ \Psi \longrightarrow bn\ \alpha\ \sharp*\ P \longrightarrow (\exists P'.\ \Psi \rhd P \Longrightarrow \alpha \prec P' \wedge (\Psi, P', Q') \in Rel)$

lemma *weakenSimI*[*case-names cAct*]:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Rel :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*
and Q :: (*'a*, *'b*, *'c*) *psi*
and C :: *'d*::*fs-name*

assumes *rOutput*: $\bigwedge\alpha\ Q'.\ [\![\Psi \rhd Q \longmapsto\alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P]\!] \Longrightarrow \exists P'.\ \Psi \rhd P \Longrightarrow\alpha \prec P' \wedge (\Psi, P', Q') \in Rel$

shows $\Psi \rhd P \rightsquigarrow_w <Rel> Q$
⟨*proof*⟩

lemma *weakenSimWeakSim*:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Rel :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*
and Q :: (*'a*, *'b*, *'c*) *psi*

assumes $(\Psi, P, Q) \in Rel$
and *cStatImp*: $\bigwedge\Psi'\ R\ S.\ (\Psi, R, S) \in Rel \Longrightarrow \Psi \rhd R \lessapprox_w <Rel> S$
and *cSim*: $\bigwedge\Psi'\ R\ S.\ (\Psi, R, S) \in Rel \Longrightarrow \Psi \rhd R \rightsquigarrow_w <Rel'> S$
and *cExt*: $\bigwedge\Psi'\ R\ S\ \Psi'.\ (\Psi, R, S) \in Rel' \Longrightarrow (\Psi \otimes \Psi', R, S) \in Rel'$
and *cSym*: $\bigwedge\Psi'\ R\ S.\ (\Psi, R, S) \in Rel \Longrightarrow (\Psi, S, R) \in Rel$

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

lemma *weakSimWeakenSim*:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Rel :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*
and Q :: (*'a*, *'b*, *'c*) *psi*

assumes *cSim*: $\Psi \rhd P \rightsquigarrow <Rel> Q$
and *cStatEq*: $\bigwedge\Psi'\ R\ S\ \Psi''.\ [\![(\Psi', R, S) \in Rel;\ \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', R, S) \in Rel$

shows $\Psi \rhd P \rightsquigarrow_w <Rel> Q$
⟨*proof*⟩

lemma *weakenSimE*:
fixes F :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Rel :: (*'b* × (*'a*, *'b*, *'c*) *psi* × (*'a*, *'b*, *'c*) *psi*) *set*
and Q :: (*'a*, *'b*, *'c*) *psi*

assumes $\Psi \rhd P \rightsquigarrow_w <Rel> Q$

shows $\bigwedge\alpha \;\; Q'.\ [\![\Psi \rhd Q \longmapsto\!\!\!\rightarrow \alpha \prec Q';\ bn\ \alpha \sharp* \Psi;\ bn\ \alpha \sharp* P]\!] \Longrightarrow$
$\exists P'.\ \Psi \rhd P \Longrightarrow \alpha \prec P' \wedge (\Psi,\ P',\ Q') \in Rel$
⟨*proof*⟩

lemma *weakenSimMonotonic*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $A :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $B :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes $\Psi \rhd P \rightsquigarrow_w <A> Q$
and $A \subseteq B$

shows $\Psi \rhd P \rightsquigarrow_w <B> Q$
⟨*proof*⟩

end

end

theory *Weaken-Bisimulation*
imports *Weaken-Simulation Weaken-Stat-Imp*
begin

context *weak*
begin

lemma *weakenMonoCoinduct*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \rightsquigarrow_w <\{(xc,\ xb,\ xa).\ x\ xc\ xb\ xa\}> P) \longrightarrow$
$(\Psi \rhd Q \rightsquigarrow_w <\{(xb,\ xa,\ xc).\ y\ xb\ xa\ xc\}> P)$
⟨*proof*⟩

lemma *weakenMonoCoinduct2*: $\bigwedge x\ y\ xa\ xb\ xc\ P\ Q\ \Psi.$
$x \leq y \Longrightarrow$
$(\Psi \rhd Q \lesssim_w <\{(xc,\ xb,\ xa).\ x\ xc\ xb\ xa\}> P) \longrightarrow$
$(\Psi \rhd Q \lesssim_w <\{(xb,\ xa,\ xc).\ y\ xb\ xa\ xc\}> P)$
⟨*proof*⟩

coinductive-set *weakenBisim* :: $('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$
where
step: $[\![\Psi \rhd P \lesssim_w <weakenBisim> Q;\ \Psi \rhd P \rightsquigarrow_w <weakenBisim> Q;$
$\forall \Psi'.\ (\Psi \otimes \Psi',\ P,\ Q) \in weakenBisim;\ (\Psi,\ Q,\ P) \in weakenBisim]\!] \Longrightarrow (\Psi,$
$P,\ Q) \in weakenBisim$
monos *weakenMonoCoinduct weakenMonoCoinduct2*

abbreviation
weakenBisimJudge $(\langle\text{-} \rhd \text{-} \approx_w \text{-}\rangle\ [70,\ 70,\ 70]\ 65)$ **where** $\Psi \rhd P \approx_w Q \equiv (\Psi,$

P, Q) $\in$ *weakenBisim*
abbreviation
weakenBisimNilJudge (‹- $\approx_w$ -› [*70*, *70*] *65*) **where** $P \approx_w Q \equiv \mathbf{1} \rhd P \approx_w Q$

lemma *weakenBisimCoinductAux*[*consumes 1*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \rhd P \lessapprox_w <(X \cup weakenBisim)> Q) \wedge$
$(\Psi \rhd P \rightsquigarrow_w <(X \cup weakenBisim)> Q) \wedge$
$(\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X \vee (\Psi \otimes \Psi', P, Q) \in weakenBisim) \wedge$
$((\Psi, Q, P) \in X \vee (\Psi, Q, P) \in weakenBisim)$

shows $(\Psi, P, Q) \in weakenBisim$
⟨*proof*⟩

lemma *weakenBisimCoinduct*[*consumes 1*, *case-names cStatImp cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \lessapprox_w <(X \cup weakenBisim)> S$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow \Psi' \rhd R \rightsquigarrow_w <(X \cup weakenBisim)> S$
and $\bigwedge \Psi'\ R\ S\ \Psi''.\ (\Psi', R, S) \in X \Longrightarrow (\Psi' \otimes \Psi'', R, S) \in X \vee \Psi' \otimes \Psi'' \rhd R \approx_w S$
and $\bigwedge \Psi'\ R\ S.\ (\Psi', R, S) \in X \Longrightarrow (\Psi', S, R) \in X \vee \Psi' \rhd S \approx_w R$

shows $\Psi \rhd P \approx_w Q$
⟨*proof*⟩

lemma *weakenBisimWeakCoinductAux*[*consumes 1*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox_w <X> Q \wedge$
$\Psi \rhd P \rightsquigarrow_w <X> Q \wedge (\forall \Psi'.\ (\Psi \otimes \Psi', P, Q) \in X) \wedge$
$(\Psi, Q, P) \in X$

shows $\Psi \rhd P \approx_w Q$
⟨*proof*⟩

lemma *weakenBisimE*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi :: 'b$
and $\Psi' :: 'b$

assumes $\Psi \rhd P \approx_w Q$

shows $\Psi \rhd P \lessapprox_w <weakenBisim> Q$
and $\Psi \rhd P \rightsquigarrow_w <weakenBisim> Q$
and $\Psi \otimes \Psi' \rhd P \approx_w Q$
and $\Psi \rhd Q \approx_w P$
⟨*proof*⟩

lemma *weakenBisimWeakCoinduct*[*consumes 1*, *case-names cStatImp cSim cExt cSym*]:
fixes $F :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $(\Psi, P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox_w <X> Q$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow_w <X> Q$
and $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$
and $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi, Q, P) \in X$

shows $(\Psi, P, Q) \in weakenBisim$
⟨*proof*⟩

lemma *weakenBisimEqWeakBisim*[*simp*]: $weakenBisim = weakBisim$
⟨*proof*⟩

lemma *weakenTransitiveWeakCoinduct*[*case-names cStatImp cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $X :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *p*: $(\Psi, P, Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatImp*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \lessapprox_w <X> Q$
and *rSim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow_w <(\{(\Psi, P, Q) \mid \Psi\ P\ Q.\ \exists P'\ Q'.\ \Psi \rhd P \sim P' \wedge (\Psi, P', Q') \in X \wedge \Psi \rhd Q' \sim Q\})> Q$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in X \Longrightarrow (\Psi \otimes \Psi', P, Q) \in X$

and *rSym*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi,\ Q,\ P) \in X$

shows $\Psi \rhd P \approx_w Q$
⟨*proof*⟩

lemma *weakenTransitiveCoinduct*[*case-names cStatImp cSim cExt cSym*, *case-conclusion bisim step*, *consumes 2*]:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $X :: ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set$

assumes *p*: $(\Psi,\ P,\ Q) \in X$
and *Eqvt*: *eqvt X*
and *rStatImp*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \lesssim_w <(X \cup weakenBisim)> Q$
and *rSim*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow \Psi \rhd P \rightsquigarrow_w <(\{(\Psi,\ P,\ Q) \mid \Psi\ P\ Q.\ \exists P'\ Q'.\ \Psi \rhd P \sim P' \wedge (\Psi,\ P',\ Q') \in (X \cup weakenBisim) \wedge \Psi \rhd Q' \sim Q\})> Q$
and *rExt*: $\bigwedge\Psi\ P\ Q\ \Psi'.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi \otimes \Psi',\ P,\ Q) \in X \cup weakenBisim$
and *rSym*: $\bigwedge\Psi\ P\ Q.\ (\Psi,\ P,\ Q) \in X \Longrightarrow (\Psi,\ Q,\ P) \in X \cup weakenBisim$

shows $\Psi \rhd P \approx_w Q$
⟨*proof*⟩

end

end

theory *Weak-Cong-Simulation*
imports *Weak-Simulation Tau-Chain*
begin

context *env* **begin**

definition
weakCongSimulation :: $'b \Rightarrow ('a,\ 'b,\ 'c)\ psi \Rightarrow ('b \times ('a,\ 'b,\ 'c)\ psi \times ('a,\ 'b,\ 'c)\ psi)\ set \Rightarrow ('a,\ 'b,\ 'c)\ psi \Rightarrow bool$ (‹- ⊳ - ⇝«-» -› [80, 80, 80, 80] 80)
where
$\Psi \rhd P \rightsquigarrow «Rel»\ Q \equiv \forall Q'.\ \Psi \rhd Q \longmapsto\tau \prec Q' \longrightarrow (\exists P'.\ \Psi \rhd P \Longrightarrow_\tau P' \wedge (\Psi,\ P',\ Q') \in Rel)$

abbreviation
weakCongSimulationNilJudge (‹- ⇝«-» -› [80, 80, 80] 80) **where** $P \rightsquigarrow «Rel»\ Q \equiv SBottom' \rhd P \rightsquigarrow «Rel»\ Q$

lemma *weakCongSimI*[*case-names cTau*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*) *set*
and Q :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *rTau*: $\bigwedge Q'.\ \Psi \rhd Q \longmapsto \tau \prec Q' \Longrightarrow \exists P'.\ \Psi \rhd P \Longrightarrow_\tau P' \wedge (\Psi, P', Q') \in Rel$

shows $\Psi \rhd P \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow Q$
⟨*proof*⟩

lemma *weakCongSimE*:
fixes F :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*) *set*
and Q :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow Q$
and $\Psi \rhd Q \longmapsto \tau \prec Q'$

obtains P' **where** $\Psi \rhd P \Longrightarrow_\tau P'$ **and** $(\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *weakCongSimClosedAux*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*) *set*
and Q :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*

assumes *EqvtRel*: *eqvt Rel*
and *PSimQ*: $\Psi \rhd P \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow (p \cdot Q)$
⟨*proof*⟩

lemma *weakCongSimClosed*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and *Rel* :: $('b \times ('a, 'b, 'c)$ *psi* $\times ('a, 'b, 'c)$ *psi*) *set*
and Q :: $('a, 'b, 'c)$ *psi*
and p :: *name prm*

assumes *EqvtRel*: *eqvt Rel*

shows $\Psi \rhd P \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow Q \Longrightarrow (p \cdot \Psi) \rhd (p \cdot P) \rightsquigarrow\!\langle Rel \rangle\!\rightsquigarrow (p \cdot Q)$

and $P \rightsquigarrow\!\!\ll Rel\gg Q \Longrightarrow (p \cdot P) \rightsquigarrow\!\!\ll Rel\gg (p \cdot Q)$
⟨*proof*⟩

lemma *weakCongSimReflexive*:
fixes $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes $\{(\Psi, P, P) \mid \Psi\ P.\ True\} \subseteq Rel$

shows $\Psi \rhd P \rightsquigarrow\!\!\ll Rel\gg P$
⟨*proof*⟩

lemma *weakStepSimTauChain*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $Q' :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $\Psi \rhd Q \Longrightarrow_\tau Q'$
and $\Psi \rhd P \rightsquigarrow\!\!\ll Rel\gg Q$
and *Sim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in Rel \Longrightarrow \Psi \rhd P \rightsquigarrow <Rel> Q$

obtains P' **where** $\Psi \rhd P \Longrightarrow_\tau P'$ **and** $(\Psi, P', Q') \in Rel$
⟨*proof*⟩

lemma *weakCongSimTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $Rel' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $T :: ('a, 'b, 'c)\ psi$
and $Rel'' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PRelQ*: $(\Psi, P, Q) \in Rel$
and *PSimQ*: $\Psi \rhd P \rightsquigarrow\!\!\ll Rel\gg Q$
and *QSimR*: $\Psi \rhd Q \rightsquigarrow\!\!\ll Rel'\gg R$
and *Set*: $\{(\Psi, P, R) \mid \Psi\ P\ R.\ \exists Q.\ (\Psi, P, Q) \in Rel \wedge (\Psi, Q, R) \in Rel'\} \subseteq Rel''$
and *Sim*: $\bigwedge \Psi\ P\ Q.\ (\Psi, P, Q) \in Rel \Longrightarrow \Psi \rhd P \rightsquigarrow <Rel> Q$

shows $\Psi \rhd P \rightsquigarrow\!\!\ll Rel''\gg R$
⟨*proof*⟩

lemma *weakCongSimStatEq*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $\Psi' :: 'b$

assumes *PSimQ*: $\Psi \rhd P \rightsquigarrow\!\!\ll Rel \gg Q$
and $\Psi \simeq \Psi'$
and *C1*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ [\![(\Psi, P, Q) \in Rel; \Psi \simeq \Psi']\!] \Longrightarrow (\Psi', P, Q) \in Rel'$

shows $\Psi' \rhd P \rightsquigarrow\!\!\ll Rel' \gg Q$
⟨*proof*⟩

lemma *weakCongSimMonotonic*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $A :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $B :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $\Psi \rhd P \rightsquigarrow\!\!\ll A \gg Q$
and $A \subseteq B$

shows $\Psi \rhd P \rightsquigarrow\!\!\ll B \gg Q$
⟨*proof*⟩

lemma *strongSimWeakCongSim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$

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

shows $\Psi \rhd P \rightsquigarrow\!\!\ll Rel' \gg Q$
⟨*proof*⟩

end

end

theory *Weak-Psi-Congruence*
imports *Weak-Cong-Simulation Weak-Bisimulation*
begin

context *env* **begin**

definition *weakPsiCongruence* :: $'b \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow bool$ (‹- ⊳ - ≐ -› [70, 70, 70] 65)
where

$\Psi \rhd P \doteq Q \equiv \Psi \rhd P \approx Q \wedge \Psi \rhd P \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ Q \wedge \Psi \rhd Q \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ P$

abbreviation
weakPsiCongNilJudge ($\langle$- $\doteq$ -$\rangle$ [70, 70] 65) **where** $P \doteq Q \equiv \mathbf{1} \rhd P \doteq Q$

lemma *weakPsiCongSym*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

assumes $\Psi \rhd P \doteq Q$

shows $\Psi \rhd Q \doteq P$
⟨*proof*⟩

lemma *weakPsiCongE*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and Ψ' :: $'b$

assumes $\Psi \rhd P \doteq Q$

shows $\Psi \rhd P \approx Q$
and $\Psi \rhd P \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ Q$
and $\Psi \rhd Q \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ P$
⟨*proof*⟩

lemma *weakPsiCongI*[*case-names cWeakBisim cSimLeft cSimRight*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and Ψ' :: $'b$

assumes $\Psi \rhd P \approx Q$
and $\Psi \rhd P \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ Q$
and $\Psi \rhd Q \rightsquigarrow\!\!\!\!\rightarrow\!\!\!\!\rightarrow \langle\!\langle weakBisim \rangle\!\rangle\ P$

shows $\Psi \rhd P \doteq Q$
⟨*proof*⟩

lemma *weakPsiCongSymI*[*consumes 1, case-names cSym cWeakBisim cSim*]:
fixes Ψ :: $'b$
and P :: $'d$::*fs-name*
and Q :: $'d$
and Ψ' :: $'b$

assumes *Prop P Q*
and $\bigwedge P\ Q.\ \textit{Prop}\ P\ Q \Longrightarrow \textit{Prop}\ Q\ P$

and $\bigwedge P\ Q.\ \textit{Prop}\ P\ Q \Longrightarrow \Psi \rhd (C\ P) \approx (C\ Q)$

and $\bigwedge P\ Q.\ \textit{Prop}\ P\ Q \Longrightarrow \Psi \rhd (C\ P) \leadsto$«*weakBisim*» $(C\ Q)$

shows $\Psi \rhd (C\ P) \doteq (C\ Q)$
⟨*proof*⟩

lemma *weakPsiCongSym2*[*consumes 1*, *case-names cWeakBisim cSim*]:
fixes Ψ :: *'b*
and Ψ' :: *'b*

assumes $\Psi \rhd P \doteq Q$

and $\bigwedge P\ Q.\ \Psi \rhd P \doteq Q \Longrightarrow \Psi \rhd (C\ P) \approx (C\ Q)$

and $\bigwedge P\ Q.\ \Psi \rhd P \doteq Q \Longrightarrow \Psi \rhd (C\ P) \leadsto$«*weakBisim*» $(C\ Q)$

shows $\Psi \rhd (C\ P) \doteq (C\ Q)$
⟨*proof*⟩

lemma *statEqWeakCong*:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Q :: (*'a*, *'b*, *'c*) *psi*
and Ψ' :: *'b*

assumes $\Psi \rhd P \doteq Q$
and $\Psi \simeq \Psi'$

shows $\Psi' \rhd P \doteq Q$
⟨*proof*⟩

lemma *weakPsiCongReflexive*:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*

shows $\Psi \rhd P \doteq P$
⟨*proof*⟩

lemma *weakPsiCongClosed*:
fixes Ψ :: *'b*
and P :: (*'a*, *'b*, *'c*) *psi*
and Q :: (*'a*, *'b*, *'c*) *psi*
and p :: *name prm*

assumes $\Psi \rhd P \doteq Q$

shows $(p \cdot \Psi) \rhd (p \cdot P) \doteq (p \cdot Q)$
⟨*proof*⟩

lemma *weakPsiCongTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \doteq Q$
and $\Psi \rhd Q \doteq R$

shows $\Psi \rhd P \doteq R$
⟨*proof*⟩

lemma *strongBisimWeakPsiCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \sim Q$

shows $\Psi \rhd P \doteq Q$
⟨*proof*⟩

lemma *structCongWeakPsiCong*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \equiv_s Q$

shows $P \doteq Q$
⟨*proof*⟩

end

end

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

context *env* **begin**

lemma *caseWeakSimPres*:
fixes $\Psi :: 'b$
and $CsP :: ('c \times ('a, 'b, 'c)\ psi)\ list$

and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $CsQ :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $M :: 'a$
and $N :: 'a$

assumes $PRelQ$: $\bigwedge\varphi\ Q.\ (\varphi, Q)\ mem\ CsQ \Longrightarrow \exists P.\ (\varphi, P)\ mem\ CsP \wedge guarded\ P \wedge Eq\ \Psi\ P\ Q$
and Sim: $\bigwedge\Psi'\ P\ Q.\ (\Psi', P, Q) \in Rel \Longrightarrow \Psi' \rhd P \rightsquigarrow <Rel>\ Q$
and $EqRel$: $\bigwedge\Psi'\ P\ Q.\ Eq\ \Psi'\ P\ Q \Longrightarrow (\Psi', P, Q) \in Rel$
and $EqSim$: $\bigwedge\Psi'\ P\ Q.\ Eq\ \Psi'\ P\ Q \Longrightarrow \Psi' \rhd P \rightsquigarrow «Rel»\ Q$

shows $\Psi \rhd Cases\ CsP \rightsquigarrow <Rel>\ Cases\ CsQ$
⟨*proof*⟩

lemma *weakCongSimCasePres*:
fixes $\Psi :: 'b$
and $CsP :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $CsQ :: ('c \times ('a, 'b, 'c)\ psi)\ list$
and $M :: 'a$
and $N :: 'a$

assumes $PRelQ$: $\bigwedge\varphi\ Q.\ (\varphi, Q)\ mem\ CsQ \Longrightarrow \exists P.\ (\varphi, P)\ mem\ CsP \wedge guarded\ P \wedge Eq\ \Psi\ P\ Q$
and $EqSim$: $\bigwedge\Psi'\ P\ Q.\ Eq\ \Psi'\ P\ Q \Longrightarrow \Psi' \rhd P \rightsquigarrow «Rel»\ Q$

shows $\Psi \rhd Cases\ CsP \rightsquigarrow «Rel»\ Cases\ CsQ$
⟨*proof*⟩

lemma *weakCongSimResPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $x :: name$
and $Rel' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $PSimQ$: $\Psi \rhd P \rightsquigarrow «Rel»\ Q$
and $eqvt\ Rel'$
and $x \sharp \Psi$
and $Rel \subseteq Rel'$
and $C1$: $\bigwedge\Psi'\ R\ S\ x.\ [\![(\Psi', R, S) \in Rel; x \sharp \Psi]\!] \Longrightarrow (\Psi', (\!|\nu x|\!)R, (\!|\nu x|\!)S) \in Rel'$

shows $\Psi \rhd (\!|\nu x|\!)P \rightsquigarrow «Rel'»\ (\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *weakCongSimResChainPres*:
fixes $\Psi :: 'b$

and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$

assumes $PSimQ$: $\Psi \rhd P \rightsquigarrow\!\!\ll Rel \gg Q$
and $eqvt\ Rel$
and $xvec \sharp* \Psi$
and $C1$: $\bigwedge \Psi'\ R\ S\ xvec.\ [\![(\Psi', R, S) \in Rel; xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!) R, (\!|\nu * xvec|\!) S) \in Rel$

shows $\Psi \rhd (\!|\nu * xvec|\!) P \rightsquigarrow\!\!\ll Rel \gg (\!|\nu * xvec|\!) Q$
⟨*proof*⟩

lemma *weakCongSimParPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$
and $Rel' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes $PSimQ$: $\bigwedge \Psi'.\ \Psi' \rhd P \rightsquigarrow\!\!\ll Rel \gg Q$
and $PSimQ'$: $\bigwedge \Psi'.\ \Psi' \rhd P \rightsquigarrow\!\!< Rel > Q$
and $StatImp$: $\bigwedge \Psi'.\ \Psi' \rhd Q \lesssim\!\!< Rel > P$

and $eqvt\ Rel$
and $eqvt\ Rel'$

and Sym: $\bigwedge \Psi'\ S\ T.\ [\![(\Psi', S, T) \in Rel]\!] \Longrightarrow (\Psi', T, S) \in Rel$
and Ext: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel]\!] \Longrightarrow (\Psi' \otimes \Psi'', S, T) \in Rel$

and $C1$: $\bigwedge \Psi'\ S\ T\ A_U\ \Psi_U\ U.\ [\![(\Psi' \otimes \Psi_U, S, T) \in Rel; extractFrame\ U = \langle A_U, \Psi_U \rangle; A_U \sharp* \Psi'; A_U \sharp* S; A_U \sharp* T]\!] \Longrightarrow (\Psi', S \parallel U, T \parallel U) \in Rel'$
and $C2$: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi', S, T) \in Rel'; xvec \sharp* \Psi']\!] \Longrightarrow (\Psi', (\!|\nu * xvec|\!) S, (\!|\nu * xvec|\!) T) \in Rel'$
and $C3$: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi', S, T) \in Rel; \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'', S, T) \in Rel$

shows $\Psi \rhd P \parallel R \rightsquigarrow\!\!\ll Rel' \gg Q \parallel R$
⟨*proof*⟩
unbundle *no relcomp-syntax*

lemma *weakCongSimBangPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $Rel :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Rel' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$
and $Rel'' :: ('b \times ('a, 'b, 'c)\ psi \times ('a, 'b, 'c)\ psi)\ set$

assumes *PEqQ*: $Eq\ P\ Q$
and *PRelQ*: $(\Psi,\ P,\ Q) \in Rel$
and $guarded\ P$
and $guarded\ Q$
and $Rel'Rel$: $Rel' \subseteq Rel$
and *FrameParPres*: $\bigwedge \Psi'\ \Psi_U\ S\ T\ U\ A_U.\ [\![(\Psi' \otimes \Psi_U,\ S,\ T) \in Rel;\ extractFrame\ U = \langle A_U,\ \Psi_U\rangle;\ A_U \sharp* \Psi';\ A_U \sharp* S;\ A_U \sharp* T]\!] \Longrightarrow (\Psi',\ U \parallel S,\ U \parallel T) \in Rel$
and *C1*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi',\ S,\ T) \in Rel;\ guarded\ S;\ guarded\ T]\!] \Longrightarrow (\Psi',\ U \parallel !S,\ U \parallel !T) \in Rel''$
and *Closed*: $\bigwedge \Psi'\ S\ T\ p.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow ((p::name\ prm) \cdot \Psi',\ p \cdot S,\ p \cdot T) \in Rel$
and *Closed'*: $\bigwedge \Psi'\ S\ T\ p.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow ((p::name\ prm) \cdot \Psi',\ p \cdot S,\ p \cdot T) \in Rel'$
and *StatEq*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi',\ S,\ T) \in Rel;\ \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'',\ S,\ T) \in Rel$
and *StatEq'*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ [\![(\Psi',\ S,\ T) \in Rel';\ \Psi' \simeq \Psi'']\!] \Longrightarrow (\Psi'',\ S,\ T) \in Rel'$
and *Trans*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi',\ S,\ T) \in Rel;\ (\Psi',\ T,\ U) \in Rel]\!] \Longrightarrow (\Psi',\ S,\ U) \in Rel$
and *Trans'*: $\bigwedge \Psi'\ S\ T\ U.\ [\![(\Psi',\ S,\ T) \in Rel';\ (\Psi',\ T,\ U) \in Rel']\!] \Longrightarrow (\Psi',\ S,\ U) \in Rel'$
and *EqSim*: $\bigwedge \Psi'\ S\ T.\ Eq\ S\ T \Longrightarrow \Psi' \rhd S \rightsquigarrow\!\!\ll Rel\gg\ T$
and *cSim*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow \Psi' \rhd S \rightsquigarrow\!\!< Rel >\ T$
and *cSym*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi',\ T,\ S) \in Rel$
and *cSym'*: $\bigwedge \Psi'\ S\ T.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi',\ T,\ S) \in Rel'$
and *cExt*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi' \otimes \Psi'',\ S,\ T) \in Rel$
and *cExt'*: $\bigwedge \Psi'\ S\ T\ \Psi''.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi' \otimes \Psi'',\ S,\ T) \in Rel'$
and *ParPres*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S,\ T) \in Rel \Longrightarrow (\Psi',\ S \parallel U,\ T \parallel U) \in Rel$
and *ParPres'*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S,\ T) \in Rel' \Longrightarrow (\Psi',\ U \parallel S,\ U \parallel T) \in Rel'$
and *ParPres2*: $\bigwedge \Psi'\ S\ T.\ Eq\ S\ T \Longrightarrow Eq\ (S \parallel S)\ (T \parallel T)$
and *ResPres*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi',\ S,\ T) \in Rel;\ xvec \sharp* \Psi']\!] \Longrightarrow (\Psi',\ (\!|\nu* xvec|\!)S,\ (\!|\nu* xvec|\!)\ T) \in Rel$
and *ResPres'*: $\bigwedge \Psi'\ S\ T\ xvec.\ [\![(\Psi',\ S,\ T) \in Rel';\ xvec \sharp* \Psi']\!] \Longrightarrow (\Psi',\ (\!|\nu* xvec|\!)S,\ (\!|\nu* xvec|\!)\ T) \in Rel'$
and *Assoc*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S \parallel (T \parallel U),\ (S \parallel T) \parallel U) \in Rel$
and *Assoc'*: $\bigwedge \Psi'\ S\ T\ U.\ (\Psi',\ S \parallel (T \parallel U),\ (S \parallel T) \parallel U) \in Rel'$
and *ScopeExt*: $\bigwedge xvec\ \Psi'\ T\ S.\ [\![xvec \sharp* \Psi';\ xvec \sharp* T]\!] \Longrightarrow (\Psi',\ (\!|\nu* xvec|\!)(S \parallel T),\ ((\!|\nu* xvec|\!)S) \parallel T) \in Rel$
and *ScopeExt'*: $\bigwedge xvec\ \Psi'\ T\ S.\ [\![xvec \sharp* \Psi';\ xvec \sharp* T]\!] \Longrightarrow (\Psi',\ (\!|\nu* xvec|\!)(S \parallel T),\ ((\!|\nu* xvec|\!)S) \parallel T) \in Rel'$
and *Compose*: $\bigwedge \Psi'\ S\ T\ U\ O.\ [\![(\Psi',\ S,\ T) \in Rel;\ (\Psi',\ T,\ U) \in Rel'';\ (\Psi',\ U,\ O) \in Rel']\!] \Longrightarrow (\Psi',\ S,\ O) \in Rel''$
and *rBangActE*: $\bigwedge \Psi'\ S\ \alpha\ S'.\ [\![\Psi' \rhd !S \longmapsto \alpha \prec S';\ guarded\ S;\ bn\ \alpha \sharp* S;\ \alpha \neq \tau;\ bn\ \alpha \sharp* subject\ \alpha]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd S \longmapsto \alpha \prec T \wedge (\mathbf{1},\ S',\ T \parallel !S) \in Rel'$
and *rBangTauE*: $\bigwedge \Psi'\ S\ S'.\ [\![\Psi' \rhd !S \longmapsto \tau \prec S';\ guarded\ S]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd S \parallel S \longmapsto \tau \prec T \wedge (\mathbf{1},\ S',\ T \parallel !S) \in Rel'$
and *rBangTauI*: $\bigwedge \Psi'\ S\ S'.\ [\![\Psi' \rhd S \parallel S \Longrightarrow_\tau S';\ guarded\ S]\!] \Longrightarrow \exists\ T.\ \Psi' \rhd$

$!S \Longrightarrow_\tau T \wedge (\Psi', T, S' \parallel !S) \in Rel'$
shows $\Psi \rhd R \parallel !P \rightsquigarrow\!\!\ll Rel''\!\!\gg R \parallel !Q$
⟨*proof*⟩
unbundle *relcomp-syntax*

end

end

theory *Weak-Cong-Pres*
imports *Weak-Psi-Congruence Weak-Cong-Sim-Pres Weak-Bisim-Pres*
begin

context *env* **begin**

lemma *weakPsiCongInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $\bigwedge Tvec.$ *length xvec* = *length Tvec* $\Longrightarrow \Psi \rhd P[xvec::=Tvec] \approx Q[xvec::=Tvec]$

shows $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \doteq M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakPsiCongOutputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $\Psi \rhd P \doteq Q$

shows $\Psi \rhd M\langle N\rangle.P \doteq M\langle N\rangle.Q$
⟨*proof*⟩

lemma *weakBisimCasePres*:
fixes Ψ :: $'b$
and *CsP* :: $('c \times ('a, 'b, 'c)$ *psi*$)$ *list*
and *CsQ* :: $('c \times ('a, 'b, 'c)$ *psi*$)$ *list*

assumes *A*: $\bigwedge\varphi\ P.\ (\varphi, P)$ *mem CsP* $\Longrightarrow \exists\, Q.\ (\varphi, Q)$ *mem CsQ* $\wedge$ *guarded Q* $\wedge$
$(\forall\, \Psi.\ \Psi \rhd P \doteq Q)$
and *B*: $\bigwedge\varphi\ Q.\ (\varphi, Q)$ *mem CsQ* $\Longrightarrow \exists\, P.\ (\varphi, P)$ *mem CsP* $\wedge$ *guarded P* $\wedge$

$(\forall\,\Psi.\ \Psi \rhd P \doteq Q)$

shows $\Psi \rhd \mathit{Cases}\ \mathit{CsP} \approx \mathit{Cases}\ \mathit{CsQ}$
⟨*proof*⟩

lemma *weakPsiCongCasePres*:
fixes Ψ :: $'b$
and CsP :: $('c \times ('a, 'b, 'c)\ \mathit{psi})\ \mathit{list}$
and CsQ :: $('c \times ('a, 'b, 'c)\ \mathit{psi})\ \mathit{list}$

assumes *A*: $\bigwedge\varphi\ P.\ (\varphi, P)\ \mathit{mem}\ \mathit{CsP} \Longrightarrow \exists\, Q.\ (\varphi, Q)\ \mathit{mem}\ \mathit{CsQ} \wedge \mathit{guarded}\ Q \wedge (\forall\,\Psi.\ \Psi \rhd P \doteq Q)$
and *B*: $\bigwedge\varphi\ Q.\ (\varphi, Q)\ \mathit{mem}\ \mathit{CsQ} \Longrightarrow \exists\, P.\ (\varphi, P)\ \mathit{mem}\ \mathit{CsP} \wedge \mathit{guarded}\ P \wedge (\forall\,\Psi.\ \Psi \rhd P \doteq Q)$

shows $\Psi \rhd \mathit{Cases}\ \mathit{CsP} \doteq \mathit{Cases}\ \mathit{CsQ}$
⟨*proof*⟩

lemma *weakPsiCongResPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ \mathit{psi}$
and Q :: $('a, 'b, 'c)\ \mathit{psi}$
and x :: *name*

assumes $\Psi \rhd P \doteq Q$
and $x \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!)P \doteq (\!|\nu x|\!)Q$
⟨*proof*⟩

lemma *weakPsiCongResChainPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ \mathit{psi}$
and Q :: $('a, 'b, 'c)\ \mathit{psi}$
and xvec :: *name list*

assumes $\Psi \rhd P \doteq Q$
and $\mathit{xvec} \sharp* \Psi$

shows $\Psi \rhd (\!|\nu* \mathit{xvec}|\!)P \doteq (\!|\nu* \mathit{xvec}|\!)Q$
⟨*proof*⟩

lemma *weakPsiCongParPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ \mathit{psi}$
and Q :: $('a, 'b, 'c)\ \mathit{psi}$
and R :: $('a, 'b, 'c)\ \mathit{psi}$

assumes $\forall\,\Psi.\ \Psi \rhd P \doteq Q$

shows $\Psi \rhd P \parallel R \doteq Q \parallel R$
⟨*proof*⟩

end

end

theory *Weak-Cong-Struct-Cong*
imports *Weak-Cong-Pres Weak-Bisim-Struct-Cong*
begin

context *env* **begin**

lemma *weakPsiCongParComm*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel Q \doteq Q \parallel P$
⟨*proof*⟩

lemma *weakPsiCongResComm*:
fixes $x :: name$
and $\Psi :: 'b$
and $y :: name$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $y \sharp \Psi$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu y|\!)P) \doteq (\!|\nu y|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *weakPsiCongResComm'*:
fixes $x :: name$
and $\Psi :: 'b$
and $xvec :: name\ list$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $xvec \sharp* \Psi$

shows $\Psi \rhd (\!|\nu x|\!)((\!|\nu * xvec|\!)P) \doteq (\!|\nu * xvec|\!)((\!|\nu x|\!)P)$
⟨*proof*⟩

lemma *weakPsiCongScopeExt*:
fixes $x :: name$
and $\Psi :: 'b$

and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp P$

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

lemma *weakPsiCongScopeExtChain*:
fixes $xvec :: name\ list$
and $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* P$

shows $\Psi \rhd (\!|\nu* xvec|\!)(P \parallel Q) \doteq P \parallel ((\!|\nu* xvec|\!) Q)$
⟨*proof*⟩

lemma *weakPsiCongParAssoc*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd (P \parallel Q) \parallel R \doteq P \parallel (Q \parallel R)$
⟨*proof*⟩

lemma *weakPsiCongParNil*:
fixes $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \parallel \mathbf{0} \doteq P$
⟨*proof*⟩

lemma *weakPsiCongResNil*:
fixes $x :: name$
and $\Psi :: 'b$

assumes $x \sharp \Psi$

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

lemma *weakPsiCongOutputPushRes*:
fixes $x :: name$
and $\Psi :: 'b$
and $M :: 'a$

and $N :: 'a$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp M$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M\langle N\rangle.P) \doteq M\langle N\rangle.(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakPsiCongInputPushRes*:
fixes x :: *name*
and Ψ :: $'b$
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P :: $('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp M$
and $x \sharp xvec$
and $x \sharp N$

shows $\Psi \rhd (\!|\nu x|\!)(M(\!|\lambda * xvec\ N|\!).P) \doteq M(\!|\lambda * xvec\ N|\!).(\!|\nu x|\!)P$
⟨*proof*⟩

lemma *weakPsiCongCasePushRes*:
fixes x :: *name*
and Ψ :: $'b$
and *Cs* :: $('c \times ('a, 'b, 'c)\ psi)\ list$

assumes $x \sharp \Psi$
and $x \sharp (map\ fst\ Cs)$

shows $\Psi \rhd (\!|\nu x|\!)(Cases\ Cs) \doteq Cases(map\ (\lambda(\varphi, P).\ (\varphi, (\!|\nu x|\!)P))\ Cs)$
⟨*proof*⟩

lemma *weakBangExt*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

assumes *guarded P*

shows $\Psi \rhd\ !P \doteq P \parallel\ !P$
⟨*proof*⟩

lemma *weakPsiCongParSym*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\forall \Psi.\ \Psi \rhd P \doteq Q$

shows $\Psi \rhd R \parallel P \doteq R \parallel Q$
⟨*proof*⟩

lemma *weakPsiCongScopeExtSym*:
fixes $x :: name$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $x \sharp \Psi$
and $x \sharp Q$

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

lemma *weakPsiCongScopeExtChainSym*:
fixes $xvec :: name\ list$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$

assumes $xvec \sharp* \Psi$
and $xvec \sharp* Q$

shows $\Psi \rhd (\!|\nu* xvec|\!)(P \parallel Q) \doteq ((\!|\nu* xvec|\!)P) \parallel Q$
⟨*proof*⟩

lemma *weakPsiCongParPresSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\bigwedge \Psi.\ \Psi \rhd P \doteq Q$

shows $\Psi \rhd R \parallel P \doteq R \parallel Q$
⟨*proof*⟩

lemma *tauCongChainBangI*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \parallel P \Longrightarrow_\tau P'$
and *guarded P*

obtains Q **where** $\Psi \rhd !P \Longrightarrow_\tau Q$ **and** $\Psi \rhd Q \sim P' \parallel !P$
⟨*proof*⟩

lemma *weakPsiCongBangPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes *PeqQ*: $\forall \Psi.\ \Psi \rhd P \doteq Q$
and *guarded P*
and *guarded Q*

shows $\Psi \rhd !P \doteq !Q$
⟨*proof*⟩

end

end

theory *Weak-Congruence*
imports *Weak-Cong-Struct-Cong Bisim-Subst*
begin

context *env* **begin**

definition *weakCongruence* :: $('a, 'b, 'c)\ psi \Rightarrow ('a, 'b, 'c)\ psi \Rightarrow bool$ (‹- $\doteq_c$ -› [70, 70] 65)
where
$P \doteq_c Q \equiv \forall \Psi\ \sigma.\ wellFormedSubst\ \sigma \longrightarrow \Psi \rhd P[<\sigma>] \doteq Q[<\sigma>]$

lemma *weakCongE*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assumes $P \doteq_c Q$
wellFormedSubst σ

shows $\Psi \rhd P[<\sigma>] \doteq Q[<\sigma>]$
⟨*proof*⟩

lemma *weakCongI*[*case-names cWeakPsiCong*]:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\bigwedge \Psi\ \sigma.\ wellFormedSubst\ \sigma \Longrightarrow \Psi \rhd P[<\sigma>] \doteq Q[<\sigma>]$

shows $P \doteq_c Q$

⟨*proof*⟩

lemma *weakCongClosed*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $p :: name\ prm$

assumes $P \doteq_c Q$

shows $(p \cdot P) \doteq_c (p \cdot Q)$
⟨*proof*⟩

lemma *weakCongReflexive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $P \doteq_c P$
⟨*proof*⟩

lemma *weakCongSym*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \doteq_c Q$

shows $Q \doteq_c P$
⟨*proof*⟩

lemma *weakCongTransitive*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $R :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \doteq Q$
and $\Psi \rhd Q \doteq R$

shows $\Psi \rhd P \doteq R$
⟨*proof*⟩

lemma *weakCongWeakBisim*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \doteq_c Q$

shows $\Psi \rhd P \approx Q$
⟨*proof*⟩

lemma *weakCongWeakPsiCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \doteq_c Q$

shows $\Psi \rhd P \doteq Q$
⟨*proof*⟩

lemma *strongBisimWeakCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \sim_s Q$

shows $P \doteq_c Q$
⟨*proof*⟩

lemma *structCongWeakCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \equiv_s Q$

shows $P \doteq_c Q$
⟨*proof*⟩

lemma *weakCongUnfold*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assumes $P \doteq_c Q$
and *wellFormedSubst* σ

shows $P[<\sigma>] \doteq_c Q[<\sigma>]$
⟨*proof*⟩

lemma *weakCongOutputPres*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

and $M :: 'a$
and $N :: 'a$

assumes $P \doteq_c Q$

shows $M\langle N\rangle.P \doteq_c M\langle N\rangle.Q$
⟨*proof*⟩

lemma *weakCongInputPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$

assumes $P \doteq_c Q$
and *distinct xvec*

shows $M(\!|\lambda * xvec\ N|\!).P \doteq_c M(\!|\lambda * xvec\ N|\!).Q$
⟨*proof*⟩

lemma *weakCongCasePresAux*:
fixes Ψ :: $'b$
and *CsP* :: $('c \times ('a, 'b, 'c)\ psi)\ list$
and *CsQ* :: $('c \times ('a, 'b, 'c)\ psi)\ list$

assumes *C1*: $\bigwedge\varphi\ P.\ (\varphi, P)\ mem\ CsP \Longrightarrow \exists\, Q.\ (\varphi, Q)\ mem\ CsQ \wedge guarded\ Q \wedge P \doteq_c Q$
and *C2*: $\bigwedge\varphi\ Q.\ (\varphi, Q)\ mem\ CsQ \Longrightarrow \exists\, P.\ (\varphi, P)\ mem\ CsP \wedge guarded\ P \wedge P \doteq_c Q$

shows *Cases CsP* $\doteq_c$ *Cases CsQ*
⟨*proof*⟩

lemma *weakCongParPres*:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and R :: $('a, 'b, 'c)$ *psi*

assumes $P \doteq_c Q$

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

lemma *weakCongResPres*:
fixes P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*

and $x :: name$

assumes $P \doteq_c Q$

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

lemma *weakCongBangPres*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $P \doteq_c Q$
and *guarded* P
and *guarded* Q

shows $!P \doteq_c !Q$
⟨*proof*⟩

end

end

theory *Tau*
imports *Weak-Congruence Bisim-Struct-Cong*
begin

locale *tau* = *env* +
fixes *nameTerm* :: *name* ⇒ $'a$

assumes *ntEqvt*[*eqvt*]: $(p::name\ prm) \cdot (nameTerm\ x) = nameTerm(p \cdot x)$
and *ntSupp*: $supp(nameTerm\ x) = \{x\}$
and *ntEq*: $\Psi \vdash (nameTerm\ x) \leftrightarrow M = (M = nameTerm\ x)$
and *subst4*: $[\![length\ xvec = length\ Tvec;\ distinct\ xvec;\ xvec\ \sharp*\ (M::'a)]\!] \Longrightarrow M[xvec::=Tvec] = M$
begin

lemma *ntChanEq*[*simp*]:
fixes $\Psi :: 'b$
and $x :: name$

shows $\Psi \vdash (nameTerm\ x) \leftrightarrow (nameTerm\ x)$
⟨*proof*⟩

lemma *nameTermFresh*[*simp*]:
fixes $x :: name$
and $y :: name$

shows $x\ \sharp\ (nameTerm\ y) = (x \neq y)$
⟨*proof*⟩

lemma *nameTermFreshChain*[*simp*]:
  fixes *xvec* :: *name list*
  and *x* :: *name*

  shows *xvec* $\sharp*$ (*nameTerm x*) = *x* $\sharp$ *xvec*
⟨*proof*⟩

definition *tauPrefix* :: ('*a*, '*b*, '*c*) *psi* ⇒ ('*a*, '*b*, '*c*) *psi* (‹τ.-› [*85*] *85*)
  where *tauPrefix P* ≡ *THE P'*. ∃*x*::*name*. *x* $\sharp$ *P* ∧ *P'* = ⦇ν*x*⦈(((*nameTerm x*)⦇$\lambda*$([]) (*nameTerm x*)⦈.**0**) ∥ ((*nameTerm x*)⟨(*nameTerm x*)⟩.*P*))

lemma *tauActionUnfold*:
  fixes *P* :: ('*a*, '*b*, '*c*) *psi*
  and *C* :: '*d*::*fs-name*

  obtains *x*::*name* **where** *x* $\sharp$ *P* **and** *x* $\sharp$ *C* **and** τ.(*P*) = ⦇ν*x*⦈(((*nameTerm x*)⦇$\lambda*$([]) (*nameTerm x*)⦈.**0**) ∥ ((*nameTerm x*)⟨(*nameTerm x*)⟩.*P*))
⟨*proof*⟩

lemma *tauActionI*:
  fixes *P* :: ('*a*, '*b*, '*c*) *psi*

  shows ∃*P'*. Ψ ▷ τ.(*P*) ⟼τ ≺ *P'* ∧ Ψ ▷ *P* ∼ *P'*
⟨*proof*⟩

lemma *outputEmpy*[*dest*]:
  assumes Ψ ▷ *M*⟨*N*⟩.*P* ⟼*K*⦇$\nu*$*xvec*⦈⟨*N'*⟩ ≺ *P'*

  shows *xvec* = []
⟨*proof*⟩

lemma *tauActionE*:
  fixes *P* :: ('*a*, '*b*, '*c*) *psi*

  assumes Ψ ▷ τ.(*P*) ⟼τ ≺ *P'*

  shows Ψ ▷ *P* ∼ *P'* **and** *supp P'* = ((*supp P*)::*name set*)
⟨*proof*⟩

lemma *tauActionEqvt*[*eqvt*]:
  fixes *P* :: ('*a*, '*b*, '*c*) *psi*
  and *p* :: *name prm*

  shows (*p* ∙ τ.(*P*)) = τ.(*p* ∙ *P*)
⟨*proof*⟩

lemma *resCases'*[*consumes 7*, *case-names cOpen cRes*]:
  fixes Ψ :: '*b*

and x :: *name*
and P :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and P' :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd (\!|\nu x|\!)P \longmapsto x\alpha \prec xP'$
and $x \sharp \Psi$
and $x \sharp x\alpha$
and $x \sharp xP'$
and *bn* $x\alpha \sharp* \Psi$
and *bn* $x\alpha \sharp* P$
and *bn* $x\alpha \sharp* $ *subject* $x\alpha$
and *rOpen*: $\bigwedge M$ *xvec yvec y N P'*. $[\![\Psi \rhd P \longmapsto M(\!|\nu*(xvec@yvec)|\!)\langle([(x, y)] \cdot N)\rangle \prec ([(x, y)] \cdot P'); y \in supp\ N;$
$x \sharp N; x \sharp P'; x \neq y; y \sharp xvec; y \sharp yvec; y \sharp M;$ *distinct xvec*; *distinct yvec*;
$xvec \sharp* \Psi; y \sharp \Psi; yvec \sharp* \Psi; xvec \sharp* P; y \sharp P;$ *yvec* $\sharp* P$; *xvec* $\sharp* M$; $y \sharp M$;
$yvec \sharp* M$; *xvec* $\sharp* yvec$; $x\alpha = M(\!|\nu*(xvec@y\#yvec)|\!)\langle N\rangle;$ $xP' = P']\!] \Longrightarrow$
Prop
and *rScope*: $\bigwedge P'$. $[\![\Psi \rhd P \longmapsto x\alpha \prec P'; xP' = (\!|\nu x|\!)P']\!] \Longrightarrow$ *Prop*

shows *Prop*
⟨*proof*⟩

lemma *parCases'*[*consumes 5*, *case-names cPar1 cPar2 cComm1 cComm2*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)$ *psi*
and Q :: $('a, 'b, 'c)$ *psi*
and α :: $'a$ *action*
and T :: $('a, 'b, 'c)$ *psi*
and C :: $'d$::*fs-name*

assumes *Trans*: $\Psi \rhd P \parallel Q \longmapsto x\alpha \prec xT$
and *bn* $x\alpha \sharp* \Psi$
and *bn* $x\alpha \sharp* P$
and *bn* $x\alpha \sharp* Q$
and *bn* $x\alpha \sharp* $ *subject* $x\alpha$
and *rPar1*: $\bigwedge P'$ A_Q Ψ_Q. $[\![\Psi \otimes \Psi_Q \rhd P \longmapsto x\alpha \prec P';$ *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$A_Q \sharp* \Psi; A_Q \sharp* P; A_Q \sharp* Q; A_Q \sharp* x\alpha; A_Q \sharp* P'; A_Q \sharp* C; xT = P' \parallel Q]\!] \Longrightarrow$ *Prop*
and *rPar2*: $\bigwedge Q'$ A_P Ψ_P. $[\![\Psi \otimes \Psi_P \rhd Q \longmapsto x\alpha \prec Q';$ *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$A_P \sharp* \Psi; A_P \sharp* P; A_P \sharp* Q; A_P \sharp* x\alpha; A_P \sharp* Q'; A_P \sharp* C; xT = P \parallel Q']\!] \Longrightarrow$ *Prop*
and *rComm1*: $\bigwedge \Psi_Q$ *M N P'* A_P Ψ_P *K xvec Q'* A_Q.

$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|N|\!) \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|\nu * xvec|\!)\langle N\rangle \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *distinct xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* K$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* K$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$; $xvec \sharp* C$; $x\alpha{=}\tau$; $xT = (\!|\nu * xvec|\!)(P' \parallel Q')]\!] \Longrightarrow Prop$
and *rComm2*: $\bigwedge \Psi_Q\ M\ xvec\ N\ P'\ A_P\ \Psi_P\ K\ Q'\ A_Q$.
$[\![\Psi \otimes \Psi_Q \rhd P \longmapsto M(\!|\nu * xvec|\!)\langle N\rangle \prec P'$; *extractFrame* $P = \langle A_P, \Psi_P\rangle$; *distinct* A_P;
$\Psi \otimes \Psi_P \rhd Q \longmapsto K(\!|N|\!) \prec Q'$; *extractFrame* $Q = \langle A_Q, \Psi_Q\rangle$; *distinct* A_Q;
$\Psi \otimes \Psi_P \otimes \Psi_Q \vdash M \leftrightarrow K$; *distinct xvec*;
$A_P \sharp* \Psi$; $A_P \sharp* \Psi_Q$; $A_P \sharp* P$; $A_P \sharp* M$; $A_P \sharp* N$; $A_P \sharp* P'$; $A_P \sharp* Q$; $A_P \sharp* xvec$; $A_P \sharp* Q'$; $A_P \sharp* A_Q$; $A_P \sharp* C$;
$A_Q \sharp* \Psi$; $A_Q \sharp* \Psi_P$; $A_Q \sharp* P$; $A_Q \sharp* K$; $A_Q \sharp* N$; $A_Q \sharp* P'$; $A_Q \sharp* Q$; $A_Q \sharp* xvec$; $A_Q \sharp* Q'$; $A_Q \sharp* C$;
$xvec \sharp* \Psi$; $xvec \sharp* \Psi_P$; $xvec \sharp* P$; $xvec \sharp* M$; $xvec \sharp* K$; $xvec \sharp* Q$; $xvec \sharp* \Psi_Q$; $xvec \sharp* C$; $x\alpha{=}\tau$; $xT = (\!|\nu * xvec|\!)(P' \parallel Q')]\!] \Longrightarrow Prop$

shows *Prop*
⟨*proof*⟩

lemma *inputCases'*[*consumes 1*, *case-names cInput*]:
fixes Ψ :: *'b*
and M :: *'a*
and *xvec* :: *name list*
and N :: *'a*
and P :: (*'a*, *'b*, *'c*) *psi*
and α :: *'a action*
and P' :: (*'a*, *'b*, *'c*) *psi*

assumes *Trans*: $\Psi \rhd M(\!|\lambda * xvec\ N|\!).P \longmapsto \alpha \prec P'$
and *rInput*: $\bigwedge K\ Tvec.\ [\![\Psi \vdash M \leftrightarrow K$; *set xvec* $\subseteq$ *supp N*; *length xvec* = *length Tvec*; *distinct xvec*; $\alpha{=}K(\!|N[xvec::=Tvec]|\!)$; $P' = P[xvec::=Tvec]]\!] \Longrightarrow Prop\ (K(\!|N[xvec::=Tvec]|\!))\ (P[xvec::=Tvec])$

shows *Prop* α P'
⟨*proof*⟩

lemma *outputCases'*[*consumes 1*, *case-names cOutput*]:
fixes Ψ :: *'b*
and M :: *'a*
and N :: *'a*
and P :: (*'a*, *'b*, *'c*) *psi*
and α :: *'a action*
and P' :: (*'a*, *'b*, *'c*) *psi*

assumes $\Psi \rhd M\langle N\rangle.P \longmapsto \alpha \prec P'$
and $\bigwedge K.\ [\![\Psi \vdash M \leftrightarrow K;\ \mathit{subject}\ \alpha{=}\mathit{Some}\ K]\!] \Longrightarrow \mathit{Prop}\ (K\langle N\rangle)\ P$

shows $\mathit{Prop}\ \alpha\ P'$
⟨*proof*⟩

lemma *tauOutput*[*dest*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ \mathit{psi}$
and M :: $'a$
and *xvec* :: *name list*
and N :: $'a$
and P' :: $('a, 'b, 'c)\ \mathit{psi}$

assumes $\Psi \rhd \tau.(P) \longmapsto M(\!|\nu{*}\mathit{xvec}|\!)\langle N\rangle \prec P'$

shows *False*
⟨*proof*⟩

lemma *tauInput*[*dest*]:
fixes Ψ :: $'b$
and P :: $('a, 'b, 'c)\ \mathit{psi}$
and M :: $'a$
and N :: $'a$
and P' :: $('a, 'b, 'c)\ \mathit{psi}$

assumes $\Psi \rhd \tau.(P) \longmapsto M(\!|N|\!) \prec P'$

shows *False*
⟨*proof*⟩

lemma *tauPrefixFrame*:
fixes P :: $('a, 'b, 'c)\ \mathit{psi}$

shows $\mathit{extractFrame}(\tau.(P)) \simeq_F \langle\varepsilon, \mathbf{1}\rangle$
⟨*proof*⟩

lemma *insertTauAssertion*:
fixes P :: $('a, 'b, 'c)\ \mathit{psi}$
and Ψ :: $'b$

shows $\mathit{insertAssertion}\ (\mathit{extractFrame}(\tau.(P)))\ \Psi \simeq_F \langle\varepsilon, \Psi\rangle$
⟨*proof*⟩

lemma *seqSubst4*:
assumes $y \sharp \sigma$
and $\mathit{wellFormedSubst}\ \sigma$

shows *substTerm.seqSubst* (*nameTerm y*) σ = *nameTerm y*
⟨*proof*⟩

lemma *tauSeqSubst*[*simp*]:
fixes *P* :: ('*a*, '*b*, '*c*) *psi*
and σ :: (*name list* × '*a list*) *list*

assumes *wellFormedSubst* σ

shows (τ.(*P*))[<σ>] = τ.(*P*[<σ>])
⟨*proof*⟩

lemma *tauSubst*[*simp*]:
fixes *P* :: ('*a*, '*b*, '*c*) *psi*
and *xvec* :: *name list*
and *Tvec* :: '*a list*

assumes *distinct xvec*
and *length xvec* = *length Tvec*

shows (τ.(*P*))[*xvec*::=*Tvec*] = τ.(*P*[*xvec*::=*Tvec*])
⟨*proof*⟩

lemma *tauFresh*[*simp*]:
fixes *P* :: ('*a*, '*b*, '*c*) *psi*
and *x* :: *name*

shows *x* ♯ τ.(*P*) = *x* ♯ *P*
⟨*proof*⟩

lemma *tauFreshChain*[*simp*]:
fixes *P* :: ('*a*, '*b*, '*c*) *psi*
and *xvec* :: *name list*

shows *xvec* ♯* (τ.(*P*)) = (*xvec* ♯* *P*)
⟨*proof*⟩

lemma *guardedTau*:
fixes *P* :: ('*a*, '*b*, '*c*) *psi*

shows *guarded*(τ.(*P*))
⟨*proof*⟩

lemma *tauChainBisim*:
fixes Ψ :: '*b*
and *P* :: ('*a*, '*b*, '*c*) *psi*
and *P'* :: ('*a*, '*b*, '*c*) *psi*
and *P''* :: ('*a*, '*b*, '*c*) *psi*

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P'$
and $\Psi \rhd P \sim P''$

obtains P''' **where** $\Psi \rhd P'' \Longrightarrow\hat{}_\tau\ P'''$ **and** $\Psi \rhd P' \sim P'''$
⟨*proof*⟩

lemma *tauChainStepCons*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *PChain*: $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P'$

obtains P'' **where** $\Psi \rhd \tau.(P) \Longrightarrow_\tau P''$ **and** $\Psi \rhd P' \sim P''$
⟨*proof*⟩

lemma *tauChainCons*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $P' :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \Longrightarrow\hat{}_\tau\ P'$

obtains P'' **where** $\Psi \rhd \tau.(P) \Longrightarrow\hat{}_\tau\ P''$ **and** $\Psi \rhd P' \sim P''$
⟨*proof*⟩

lemma *weakTransitionTau*:
fixes $\Psi :: 'b$
and $Q :: ('a, 'b, 'c)\ psi$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ action$
and $P' :: ('a, 'b, 'c)\ psi$

assumes *PTrans*: $\Psi : Q \rhd P \Longrightarrow \alpha \prec P'$
and $bn\ \alpha \sharp* \Psi$
and $bn\ \alpha \sharp* P$

obtains P'' **where** $\Psi : Q \rhd \tau.(P) \Longrightarrow \alpha \prec P''$ **and** $\Psi \rhd P' \sim P''$
⟨*proof*⟩

end

end

theory *Sum*
imports *Semantics Close-Subst*
begin

context *env*

begin

abbreviation *sumAssertJudge* (‹- $\oplus_-$ -› [*150*, *50*, *50*] *150*)
  where (*P*::(*'a*, *'b*, *'c*) *psi*) $\oplus_\varphi$ *Q* ≡ *Cases* [(φ, *P*), (φ, *Q*)]

lemma *SumAssert1*:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *Q* :: (*'a*, *'b*, *'c*) *psi*

  assumes $\Psi \rhd P \longmapsto Rs$
  and $\Psi \vdash \varphi$
  and *guarded P*

  shows $\Psi \rhd P \oplus_\varphi Q \longmapsto Rs$
⟨*proof*⟩

lemma *SumAssert2*:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *Q* :: (*'a*, *'b*, *'c*) *psi*

  assumes $\Psi \rhd Q \longmapsto Rs$
  and $\Psi \vdash \varphi$
  and *guarded Q*

  shows $\Psi \rhd P \oplus_\varphi Q \longmapsto Rs$
⟨*proof*⟩

lemma *sumAssertCases*[*consumes 2*, *case-names cSum1 cSum2*]:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *Q* :: (*'a*, *'b*, *'c*) *psi*
  and φ :: *'c*

  assumes $\Psi \rhd P \oplus_\varphi Q \longmapsto Rs$
  and $\Psi \vdash \varphi$
  and *rSum1*: $[\![\Psi \rhd P \longmapsto Rs; \mathit{guarded}\ P]\!] \Longrightarrow \mathit{Prop}$
  and *rSum2*: $[\![\Psi \rhd Q \longmapsto Rs; \mathit{guarded}\ Q]\!] \Longrightarrow \mathit{Prop}$

  shows *Prop*
⟨*proof*⟩

lemma *sumElim*[*dest*]:
  fixes Ψ :: *'b*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *Q* :: (*'a*, *'b*, *'c*) *psi*
  and φ :: *'c*

assumes $\Psi \rhd P \oplus_\varphi Q \longmapsto Rs$
and $\neg(\Psi \vdash \varphi)$

shows *False*
⟨*proof*⟩

end

locale *sum* = *env* +
fixes $T :: 'c$

assumes *Top*: $\Psi \vdash T$
and *TopEqvt*[*eqvt*]: $((p::name\ prm) \cdot T) = T$
and *TopSubst*[*simp*]: $substCond\ T\ xvec\ Tvec = T$
begin

abbreviation *topJudge* (‹⊤› *150*) **where** $\top \equiv T$
abbreviation *sumJudge* (**infixr** ‹⊕› *80*) **where** $P \oplus Q \equiv P \oplus_\top Q$

lemma *topSeqSubst*[*simp*]:
shows $(substCond.seqSubst\ T\ \sigma) = T$
⟨*proof*⟩

lemma *suppTop*:
shows $((supp(\top))::name\ set) = \{\}$
⟨*proof*⟩

lemma *freshTop*[*simp*]:
fixes $x :: name$
and $xvec :: name\ list$
and $Xs :: name\ set$

shows $x \sharp \top$ **and** $xvec \sharp* \top$ **and** $Xs \sharp* \top$
⟨*proof*⟩

lemma *sumSubst*[*simp*]:
fixes $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $xvec :: name\ list$
and $Tvec :: 'a\ list$

assumes $length\ xvec = length\ Tvec$
and $distinct\ xvec$

shows $(P \oplus Q)[xvec::=Tvec] = (P[xvec::=Tvec] \oplus Q[xvec::=Tvec])$
⟨*proof*⟩

lemma *sumSeqSubst*[*simp*]:
fixes $P :: ('a, 'b, 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$
and $\sigma :: (name\ list \times 'a\ list)\ list$

assumes *wellFormedSubst* σ

shows $(P \oplus Q)[<\sigma>] = ((P[<\sigma>]) \oplus (Q[<\sigma>]))$
⟨*proof*⟩

lemma *Sum1*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \longmapsto Rs$
and *guarded P*

shows $\Psi \rhd P \oplus Q \longmapsto Rs$
⟨*proof*⟩

lemma *Sum2*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd Q \longmapsto Rs$
and *guarded Q*

shows $\Psi \rhd P \oplus Q \longmapsto Rs$
⟨*proof*⟩

lemma *sumCases*[*consumes 1*, *case-names cSum1 cSum2*]:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \oplus Q \longmapsto Rs$
and *rSum1*: $[\![\Psi \rhd P \longmapsto Rs; guarded\ P]\!] \Longrightarrow Prop$
and *rSum2*: $[\![\Psi \rhd Q \longmapsto Rs; guarded\ Q]\!] \Longrightarrow Prop$

shows *Prop*
⟨*proof*⟩

end

end

theory *Tau-Sim*
imports *Tau Sum*
begin

nominal-datatype *'a prefix* =
  pInput 'a::fs-name name list 'a
| *pOutput 'a 'a*
| *pTau*

context *tau*
begin

nominal-primrec *bindPrefix* :: *'a prefix* ⇒ (*'a*, *'b*, *'c*) *psi* ⇒ (*'a*, *'b*, *'c*) *psi* (‹-·-›
[*100*, *100*] *100*)
where
  bindPrefix (*pInput M xvec N*) *P* = *M*⟮λ∗*xvec N*⟯.*P*
| *bindPrefix* (*pOutput M N*) *P* = *M*⟨*N*⟩.*P*
| *bindPrefix* (*pTau*) *P* = τ.(*P*)
⟨*proof*⟩

lemma *bindPrefixEqvt*[*eqvt*]:
  fixes *p* :: *name prm*
  and α :: *'a prefix*
  and *P* :: (*'a*, *'b*, *'c*) *psi*

  shows (*p* ∙ (α·*P*)) = (*p* ∙ α)·(*p* ∙ *P*)
⟨*proof*⟩

lemma *prefixCases*[*consumes 1*, *case-names cInput cOutput cTau*]:
  fixes Ψ :: *'b*
  and α :: *'a prefix*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and β :: *'a action*
  and *P'* :: (*'a*, *'b*, *'c*) *psi*

  assumes Ψ ▷ α·*P* ⟼β ≺ *P'*
  and *rInput*: ⋀*M xvec N K Tvec*. ⟦Ψ ⊢ *M* ↔ *K*; *set xvec* ⊆ *supp N*; *length*
xvec = *length Tvec*; *distinct xvec*⟧ ⟹
                                Prop (*pInput M xvec N*) (*K*⟮*N*[*xvec*::=*Tvec*]⟯)
(*P*[*xvec*::=*Tvec*])
  and *rOutput*: ⋀*M N K*. Ψ ⊢ *M* ↔ *K* ⟹ *Prop* (*pOutput M N*) (*K*⟨*N*⟩) *P*
  and *rTau*: Ψ ▷ *P* ∼ *P'* ⟹ *Prop* (*pTau*) (τ) *P'*

  shows *Prop* α β *P'*
⟨*proof*⟩

lemma *prefixTauCases*[*consumes 1*, *case-names cTau*]:
  fixes α :: *'a prefix*
  and *P* :: (*'a*, *'b*, *'c*) *psi*
  and *P'* :: (*'a*, *'b*, *'c*) *psi*

  assumes Ψ ▷ α·*P* ⟼τ ≺ *P'*

and $rTau$: $\Psi \rhd P \sim P' \Longrightarrow Prop\ (pTau)\ P'$

shows $Prop\ \alpha\ P'$
⟨*proof*⟩

lemma *hennessySim1*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \leadsto <Rel> Q$
and $C1$: $\bigwedge \Psi\ P\ Q\ R.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel]\!] \Longrightarrow (\Psi, P, R) \in Rel$

shows $\Psi \rhd \tau.(P) \leadsto « Rel » Q$
⟨*proof*⟩

lemma *hennessySim2*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $PTrans$: $\Psi \rhd P \longmapsto \tau \prec P'$
and $P'RelQ$: $(\Psi, P', Q) \in Rel$
and $C1$: $\bigwedge \Psi\ P\ Q\ R.\ [\![(\Psi, P, Q) \in Rel; \Psi \rhd Q \sim R]\!] \Longrightarrow (\Psi, P, R) \in Rel$

shows $\Psi \rhd P \leadsto « Rel » \tau.(Q)$
⟨*proof*⟩

lemma *hennessySim3*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \leadsto <Rel> Q$
and $C1$: $\bigwedge Q'.\ \Psi \rhd Q \longmapsto \tau \prec Q' \Longrightarrow (\Psi, P, Q') \notin Rel$

shows $\Psi \rhd P \leadsto « Rel » Q$
⟨*proof*⟩

lemma *tauLaw1SimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $\Psi \rhd P \leadsto <Rel> Q$
and $eqvt\ Rel$
and $C1$: $\bigwedge \Psi\ P\ Q\ R.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel]\!] \Longrightarrow (\Psi, P, R) \in Rel$

shows $\Psi \rhd \tau.(P) \leadsto <Rel> Q$

⟨*proof*⟩

lemma *tauLaw1SimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes *eqvt Rel*
and $(\Psi, P, Q) \in Rel$
and *C1*: $\bigwedge \Psi\ P\ Q\ R.\ [\![(\Psi, P, Q) \in Rel; \Psi \rhd Q \sim R]\!] \Longrightarrow (\Psi, P, R) \in Rel$

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

lemma *tauLaw3SimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ prefix$

assumes *eqvt Rel*
and $(\Psi, P, Q) \in Rel$
and *Subst*: $\bigwedge xvec\ Tvec.\ length\ xvec = length\ Tvec \Longrightarrow (\Psi, P[xvec::=Tvec], Q[xvec::=Tvec]) \in Rel$
and *C1*: $\bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$
and *rExt*: $\bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in Rel \Longrightarrow (\Psi \otimes \Psi', P, Q) \in Rel$

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

lemma *tauLaw3SimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes *eqvt Rel*
and *Subst*: $\bigwedge \Psi\ xvec\ Tvec.\ length\ xvec = length\ Tvec \Longrightarrow (\Psi, P[xvec::=Tvec], \tau.(Q[xvec::=Tvec])) \in Rel$
and *C1*: $\bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$
and $\bigwedge \Psi.\ (\Psi, P, \tau.(Q)) \in Rel$

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

lemma *tauLaw3CongSimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

and $Q :: ('a, 'b, 'c)\ psi$

assumes $(\Psi, P, Q) \in Rel$
and $C1: \bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$
and $rExt: \bigwedge \Psi\ P\ Q\ \Psi'.\ (\Psi, P, Q) \in Rel \Longrightarrow (\Psi \otimes \Psi', P, Q) \in Rel$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P)) \rightsquigarrow\!\!\ll Rel \gg \alpha{\cdot}Q$
⟨*proof*⟩

lemma *tauLaw3CongSimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes $(\Psi, P, Q) \in Rel$
and $C1: \bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$
and $\bigwedge \Psi.\ (\Psi, P, \tau.(Q)) \in Rel$

shows $\Psi \rhd \alpha{\cdot}P \rightsquigarrow\!\!\ll Rel \gg \alpha{\cdot}(\tau.(Q))$
⟨*proof*⟩

end

locale *tauSum* = *tau* + *sum*
begin

lemma *tauLaw2SimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes $Id: \bigwedge \Psi\ P.\ (\Psi, P, P) \in Rel$
and $C1: \bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$

shows $\Psi \rhd P \oplus \tau.(P) \rightsquigarrow\!\!< Rel > \tau.(P)$
⟨*proof*⟩

lemma *tauLaw2CongSimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes $Id: \bigwedge \Psi\ P.\ (\Psi, P, P) \in Rel$
and $C1: \bigwedge \Psi\ P\ Q\ R\ S.\ [\![\Psi \rhd P \sim Q; (\Psi, Q, R) \in Rel; \Psi \rhd R \sim S]\!] \Longrightarrow (\Psi, P, S) \in Rel$

shows $\Psi \rhd P \oplus \tau.(P) \rightsquigarrow\!\!\ll Rel \gg \tau.(P)$
⟨*proof*⟩

lemma *tauLaw2SimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *C1*: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi, P, Q) \in Rel$

shows $\Psi \rhd \tau.(P) \rightsquigarrow <Rel> P \oplus \tau.(P)$
⟨*proof*⟩

lemma *tauLaw2CongSimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *C1*: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi, P, Q) \in Rel$

shows $\Psi \rhd \tau.(P) \rightsquigarrow «Rel» P \oplus \tau.(P)$
⟨*proof*⟩

lemma *tauLaw4SimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes $\bigwedge \Psi\ P.\ (\Psi, P, P) \in Rel$
and *C1*: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi, P, Q) \in Rel$

shows $\Psi \rhd \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \rightsquigarrow <Rel> \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4CongSimLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes $\bigwedge \Psi\ P.\ (\Psi, P, P) \in Rel$
and *C1*: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi, P, Q) \in Rel$

shows $\Psi \rhd \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \rightsquigarrow «Rel» \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4SimRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

and $\alpha :: 'a\ prefix$

assumes $C1$: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi,\ P,\ Q) \in Rel$
and $\bigwedge \Psi\ P.\ (\Psi,\ P,\ P) \in Rel$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P) \oplus Q) \rightsquigarrow <Rel> \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4CongSimRight*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$
and $\alpha :: 'a\ prefix$

assumes $C1$: $\bigwedge \Psi\ P\ Q.\ \Psi \rhd P \sim Q \Longrightarrow (\Psi,\ P,\ Q) \in Rel$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P) \oplus Q) \rightsquigarrow «Rel» \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

end

end

theory *Tau-Stat-Imp*
imports *Tau-Sim Weaken-Stat-Imp*
begin

locale *weakTauLaws* = *weak* + *tau*
begin

lemma *tauLaw1StatImpLeft*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$

assumes $\Psi \rhd P \lesssim_w <Rel> Q$
and $(\Psi,\ \tau.(P),\ Q) \in Rel$

shows $\Psi \rhd \tau.(P) \lesssim_w <Rel> Q$
⟨*proof*⟩

lemma *tauLaw1StatImpRight*:
fixes $\Psi :: 'b$
and $P :: ('a,\ 'b,\ 'c)\ psi$
and $Q :: ('a,\ 'b,\ 'c)\ psi$

assumes $\Psi \rhd P \lesssim <Rel> Q$
and $C1$: $\bigwedge \Psi\ P\ Q\ R.\ [\![(\Psi,\ P,\ Q) \in Rel;\ \Psi \rhd Q \sim R]\!] \Longrightarrow (\Psi,\ P,\ R) \in Rel'$

shows $\Psi \rhd P \lessapprox <Rel'> \tau.(Q)$
⟨*proof*⟩

end

context *tau* **begin**

lemma *tauLaw3StatImpLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ prefix$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', \alpha \cdot (\tau.(P)), \alpha \cdot Q) \in Rel$

shows $\Psi \rhd \alpha \cdot (\tau.(P)) \lessapprox <Rel> \alpha \cdot Q$
⟨*proof*⟩

lemma *tauLaw3StatImpRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', \alpha \cdot P, \alpha \cdot (\tau.(Q))) \in Rel$

shows $\Psi \rhd \alpha \cdot P \lessapprox <Rel> \alpha \cdot (\tau.(Q))$
⟨*proof*⟩

end

context *tauSum* **begin**

lemma *tauLaw2StatImpLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', P \oplus \tau.(P), \tau.(P)) \in Rel$

shows $\Psi \rhd P \oplus \tau.(P) \lessapprox <Rel> \tau.(P)$
⟨*proof*⟩

lemma *tauLaw2StatImpRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', \tau.(P), P \oplus \tau.(P)) \in Rel$

shows $\Psi \rhd \tau.(P) \lessapprox <Rel> P \oplus \tau.(P)$
⟨*proof*⟩

lemma *tauLaw4StatImpLeft*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $M :: 'a$
and $N :: 'a$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q), \alpha{\cdot}(\tau.(P) \oplus Q)) \in Rel$

shows $\Psi \rhd \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \lessapprox_{<Rel>} \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4StatImpRight*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $Q :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ prefix$

assumes *C1*: $\bigwedge \Psi'.\ (\Psi \otimes \Psi', \alpha{\cdot}(\tau.(P) \oplus Q), \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q)) \in Rel$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P) \oplus Q) \lessapprox_{<Rel>} \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

end

end

theory *Tau-Laws-Weak*
imports *Weaken-Bisimulation Weak-Congruence Tau-Sim Tau-Stat-Imp*
begin

context *weakTauLaws* **begin**

lemma *tauLaw1*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd \tau.(P) \approx P$
⟨*proof*⟩

lemma *tauLaw3*:
fixes $\Psi :: 'b$
and $\alpha :: 'a\ prefix$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P)) \approx \alpha{\cdot}P$
⟨*proof*⟩

lemma *tauLaw3PsiCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd \alpha{\cdot}(\tau.(P)) \doteq \alpha{\cdot}P$
⟨*proof*⟩

lemma *tauLaw3Cong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\alpha{\cdot}(\tau.(P)) \doteq_c \alpha{\cdot}P$
⟨*proof*⟩

end

end

theory *Tau-Laws-No-Weak*
imports *Tau-Sim Tau-Stat-Imp*
begin

context *tauSum*
begin

lemma *tauLaw2*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \oplus \tau.(P) \approx \tau.(P)$
⟨*proof*⟩

lemma *tauLawPsiCong2*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd P \oplus \tau.(P) \doteq \tau.(P)$
⟨*proof*⟩

lemma *tauLawCong2*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

shows $P \oplus \tau.(P) \doteq_c \tau.(P)$
⟨*proof*⟩

lemma *tauLaw4*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$

and $\alpha :: 'a\ prefix$

shows $\Psi \rhd \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \approx \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4PsiCong*:
fixes $\Psi :: 'b$
and $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ prefix$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\Psi \rhd \alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \doteq \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

lemma *tauLaw4Cong*:
fixes $P :: ('a, 'b, 'c)\ psi$
and $\alpha :: 'a\ prefix$
and $Q :: ('a, 'b, 'c)\ psi$

shows $\alpha{\cdot}P \oplus \alpha{\cdot}(\tau.(P) \oplus Q) \doteq_c \alpha{\cdot}(\tau.(P) \oplus Q)$
⟨*proof*⟩

end

end

References

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

[2] J. Bengtson, M. Johansson, J. Parrow, and B. Victor. Psi-calculi: Mobile processes, nominal data, and logic. In *LICS*, pages 39–48. IEEE Computer Society, 2009.

[3] J. Bengtson and J. Parrow. Psi-calculi in isabelle. In S. Berghofer, T. Nipkow, C. Urban, and M. Wenzel, editors, *TPHOLs*, volume 5674 of *Lecture Notes in Computer Science*, pages 99–114. Springer, 2009.

[4] M. Johansson, J. Bengtson, J. Parrow, and B. Victor. Weak equivalences in psi-calculi. In *LICS*, pages 322–331. IEEE Computer Society, 2010.